



An investigation into automated processes for generating focus maps

BRIAN SANJEEWA RUPASINGHE KALUPAHANA ARACHCHIGE

**A thesis submitted in partial fulfilment of the
requirements of the University of East London
for the degree of Doctor of Philosophy**

April 2015

Abstract

The use of geographic information for mobile applications such as wayfinding has increased rapidly, enabling users to view information on their current position in relation to the neighbouring environment. This is due to the ubiquity of small devices like mobile phones, coupled with location finding devices utilising global positioning system. However, such applications are still not attractive to users because of the difficulties in viewing and identifying the details of the immediate surroundings that help users to follow directions along a route. This results from a lack of presentation techniques to highlight the salient features (such as landmarks) among other unique features. Another problem is that since such applications do not provide any eye-catching distinction between information about the region of interest along the route and the background information, users are not tempted to focus and engage with wayfinding applications. Although several approaches have previously been attempted to solve these deficiencies by developing focus maps, such applications still need to be improved in order to provide users with a visually appealing presentation of information to assist them in wayfinding. The primary goal of this research is to investigate the processes involved in generating a visual representation that allows key features in an area of interest to stand out from the background in focus maps for wayfinding users. In order to achieve this, the automated processes in four key areas - spatial data structuring, spatial data enrichment, automatic map generalization and spatial data mining - have been thoroughly investigated by testing existing algorithms and tools. Having identified the gaps that need to be filled in these processes, the research has developed new algorithms and tools in each area through thorough testing and validation. Thus, a new triangulation data structure is developed to retrieve the adjacency relationship between polygon features required for data enrichment and automatic map generalization. Further, a new hierarchical clustering algorithm is developed to group polygon features under data enrichment required in the automatic generalization process. In addition, two generalization algorithms for polygon merging are developed for generating a generalized background for focus maps, and finally a decision tree algorithm - C4.5 - is customised for deriving salient features,

including the development of a new framework to validate derived landmark saliency in order to improve the representation of focus maps.

Table of Contents

Abstract.....	i
Table of Contents.....	iii
List of Figures	ix
List of Tables	xx
List of Abbreviations	xxiv
Acknowledgements.....	xxvi
Dedication	xxvii
Chapter 1 Introduction	1
1.1 Research motivation	1
1.2 Background	4
1.3 Outline of the thesis	9
Chapter 2 Research context	11
2.1 Focus maps.....	11
2.2 Map generalization.....	14
2.2.1 Knowledge acquisition for generalization.....	15
2.2.2 Conceptual architecture for generalization	16
2.2.3 Models of generalization	18
2.2.4 Generalization operators	19
2.2.5 Generalization algorithms.....	20
2.2.6 Review of the generalization frameworks	22
2.2.7 Relationship of generalization models to wayfinding maps	25
2.2.8 Related work on building geometries	27
2.3 Data enrichment.....	31
2.3.1 Relations in data enrichment.....	32
2.3.2 Clustering	33
2.3.3 Related work on clustering building geometries	34
2.3.4 Related work on data mining.....	40
2.4 Triangulation	41

2.4.1	Delaunay triangulation.....	42
2.4.2	Constrained Delaunay triangulation.....	44
2.4.3	Conforming Delaunay triangulation.....	46
2.4.4	Polygon triangulation.....	50
2.4.5	Related work on building geometries.....	51
2.5	Knowledge discovery	53
2.5.1	Data mining methods.....	54
2.5.2	Salient landmarks.....	56
2.5.3	Related work in deriving landmark saliency	57
2.6	Problem scope	58
2.7	Research objective.....	61
2.8	Research questions.....	61
2.9	Conclusion.....	62
Chapter 3	Research Methodology	63
3.1	Related work for the adaptation of research design.....	63
3.2	Research design	66
3.3	Methods adopted	67
3.3.1	Constrained triangulation spatial data structure.....	67
3.3.2	Spatial clustering of polygons under data enrichment.....	69
3.3.3	Automatic map generalization with building aggregation	72
3.3.4	Emphasis of salient landmarks.....	75
3.4	Conclusion.....	78
Chapter 4	Implementation - I: Spatial Data Structure.....	79
4.1	Testing of an existing constrained Delaunay triangulation algorithm	80
4.1.1	Input data structure	82
4.1.2	Evaluation of the results of constrained Delaunay triangulation	83
4.2	Testing of an existing conforming Delaunay triangulation algorithm	85
4.2.1	Input data structure	85
4.2.2	Enriching Steiner points	86
4.2.3	Evaluation of the results of conforming Delaunay triangulation.....	87
4.3	Testing of a new constrained algorithm on polygon triangulation	89
4.3.1	Input data structure	89

4.3.2	Triangulation algorithm	89
4.3.3	Evaluation of the results of constrained algorithm on polygon triangulation.....	92
4.4	Testing of a new constrained algorithm on Delaunay triangulation	92
4.4.1	Input data structure	94
4.4.2	Triangulation algorithm	94
4.4.3	Proximity links derivation between polygons.....	102
4.4.4	Evaluation of the results of the constrained algorithm on Delaunay triangulation	103
4.4.5	Validation of the constrained algorithm on Delaunay triangulation	105
4.5	Outcome of the triangulation algorithms used for testing	106
4.5.1	Constrained Delaunay triangulation	106
4.5.2	Conforming Delaunay triangulation.....	106
4.5.3	New constrained algorithm on polygon triangulation.....	107
4.5.4	New constrained algorithm on Delaunay triangulation	107
4.5.5	Comparison of the triangulation algorithms.....	107
4.6	Conclusion	110
Chapter 5	Implementation - II: Spatial Data Enrichment Process	111
5.1	Hierarchical polygon clustering process	111
5.2	Automation of hierarchical clustering process	113
5.2.1	Derivation of the Gestalt factors.....	114
5.2.2	Clustering algorithm.....	120
5.3	Testing of automatic clustering	126
5.3.1	Results of automatic clustering.....	128
5.3.2	Synopsis of the contributing algorithms used	132
5.3.3	Evaluation of the clustering results	133
5.4	Shape enrichment of clusters	150
5.4.1	Testing of algorithms for retrieving buildings at the cluster outline	150
5.4.2	Testing of new algorithms for the cluster shape enrichment.....	154
5.4.3	Synopsis of the contributing algorithms used	161
5.5	Data enrichment workflow	161
5.6	Conclusion.....	163
Chapter 6	Implementation - III: Data Mining Process	164
6.1	Data enrichment for the Data Mining process.....	165

6.1.1	Automatic derivation of attribute values.....	165
6.2	Data mining approach.....	180
6.2.1	Data pre-processing	180
6.2.2	Testing of the data mining algorithms on a synthetic data set	181
6.2.3	Testing of the data mining algorithms on a real data set	190
6.2.4	Evaluation of the results of the three data mining algorithms.....	195
6.3	Conclusion.....	198
Chapter 7 Implementation - IV: Automatic Map Generalization Process.....		199
7.1	Symbolization algorithm with squaring or enlargement.....	200
7.2	Building cluster aggregation with orthogonal sides	202
7.2.1	Creating amalgam with dilation and erosion.....	202
7.2.2	Creating amalgam with concave hull generation	205
7.2.3	Squaring edges of the amalgam.....	205
7.2.4	Testing of amalgams after squaring edges	208
7.2.5	Enlargement of narrow sections and juts	210
7.2.6	Simplification of granular edges	214
7.2.7	Testing of amalgams after complete generalization process	217
7.2.8	Modified building aggregation algorithm with orthogonal sides	222
7.2.9	Testing of amalgams with the modified algorithm.....	226
7.3	Building cluster aggregation with non-orthogonal sides	230
7.3.1	Aggregation algorithm with triangulation	230
7.3.2	Simplification algorithm.....	238
7.3.3	Testing of building cluster aggregation with non-orthogonal sides	238
7.4	Conclusion.....	244
Chapter 8 Results and discussion		245
8.1	Dealing with issues in generating the results for focus maps	246
8.1.1	Fixing issues in building clustering	246
8.1.2	Fixing issues of enrichment of building information for deriving salient landmarks	248
8.1.3	Fixing issues in the aggregation algorithms developed	251
8.1.4	Fixing issues in the simplification of aggregated amalgams	254
8.1.5	Further incorporation and development of generalization tools required for deriving focus maps.....	255

8.2	Results of focus maps	255
8.2.1	Specifications used in the focus map generation process.....	265
8.2.2	External validation of the results of generalization.....	267
8.2.3	External validation of the results of landmark saliency.....	277
8.3	Discussion	313
8.4	Conclusion.....	326
Chapter 9	Conclusions	327
9.1	Findings.....	327
9.2	Contribution	331
9.3	Future research	334
Bibliography	335
Appendices	350
A	Prototypes of Graphical User Interfaces with proprietary software	350
A.1	Input file creation for constrained Delaunay triangulation	350
A.2	Input file creation for the conforming Delaunay and the Delaunay constrained triangulations.....	351
B	Prototypes of Graphical User Interfaces with open source software	352
B.1	Constrained Delaunay triangulation.....	352
B.2	Conforming Delaunay triangulation and polygon triangulation.....	353
B.3	Constrained algorithm on Delaunay triangulation and spatial clustering with cluster shape enrichment	354
B.4	Polygon cluster matching.....	356
B.5	Spatial clustering considering the context.....	357
B.6	Generalization of building geometries.....	359
C	Existing algorithms.....	360
C.1	Prim’s algorithm for creating the Minimum Spanning Tree	360
C.2	Orientation of building polygons on the wall statistical weighting	362
C.3	Building simplification algorithm	363
D	Clustering	364
D.1	Topographic maps used for the clustering experiment: Phase I	364
D.2	Regions of the digital topographic data used for clustering.....	365
D.3	Instruction to subjects in the clustering experiment: Phase I	367

D.4	Manual clustering output of a subject from the expert group	368
D.5	Manual clustering output of a subject from the lay group	369
D.6	Automatic and manual cluster matching results	370
D.7	Questionnaire for evaluating the automatic clustering.....	392
E	SQL Queries	397
E.1	Handling PostGIS geometries in the PostgreSQL database	397
F	Data mining	400
F.1	Sensitivity analysis workflow in the WEKA software	400
F.2	Results of the sensitivity analysis in the WEKA software	401
F.3	Attribute transformation in the WEKA software	402
F.4	Data mining user interface.....	403
F.5	Enriched real test data (part of).....	404
F.6	User interface for the salient landmark evaluation	405
F.7	Results of the salient landmark evaluation at a decision point (Tower Hamlets area)	406
G	Pseudo codes of the developed algorithms.....	407
G.1	Constrained algorithm on Delaunay triangulation	407
G.2	Clustering algorithm.....	410
G.3	Cluster shape enrichment.....	420
G.4	Symbolization algorithm	423
G.5	Squaring algorithm.....	425
G.6	Enlargement algorithm	427
G.7	Simplification algorithm	432
G.8	Building aggregation with orthogonal sides	434
G.9	Building aggregation with non-orthogonal sides.....	442
H	Terminology	447

List of Figures

Figure 1.1 (a) Mobile map with symbols represented by relevance values with different opacities, from Reichenbacher (2005) and (b) traditional cartographic map.	6
Figure 1.2 The surrounding context of a mobile map user.	7
Figure 2.1 Focus maps represented with variable scales.	12
Figure 2.2 Requirement of generalization of a map.	14
Figure 2.3 (a) Relation between graphic and conceptual generalization and (b) Ladder approach left and Star approach right.	17
Figure 2.4 Generalization as a sequence of modelling operations.	18
Figure 2.5 Generalization operators defined for the AGENT project simplified after Bader <i>et al.</i> (1999).	21
Figure 2.6 AGENT generalization procedure for a single object or a group of objects.	24
Figure 2.7 Characteristics of an MRDB - storage of multiple representations of objects (left) and linkage of corresponding objects (right).	26
Figure 2.8 (a) A pair of buildings at total separation (b) displacement of building B_1 to building B_2 has created a corner touching situation and (c) rotating, aligning and merging of building B_1 with building B_2 , has created a corner touching situation.	28
Figure 2.9 A pair of buildings in a cluster at different positions: (a) total overhanging (b) partial overhanging (c) almost overhanging (d) corner touching and (e) total separation.	28
Figure 2.10 Area aggregation based on morphological operators.	29
Figure 2.11 Building aggregation.	30
Figure 2.12 Building clustering with three steps.	35
Figure 2.13 Building clustering with the MST.	37
Figure 2.14 Hierarchical relationship of constraints for building clustering.	40
Figure 2.15 (a) A domain (Ω) with a set of points and (b) triangulation of the points.	42
Figure 2.16 (a) Non-Delaunay and (b) Delaunay stable triangulation, based on Žalik (2005).	42

Figure 2.17 (a) A planar points and straight line edge graph $G(P, E)$ called PSLG (b) conventional Delaunay triangulation of point set P (c) CDT of $G(P, E)$ and (d) illustration of the modified circumcircle criterion on the hatched triangle for the CDT.	45
Figure 2.18 Execution of the CNDT algorithm on a simple example.	48
Figure 2.19 Continuation of the CNDT algorithm with splitting edges and triangles.	49
Figure 2.20 Recursive process of finding triangles in polygon triangulation.	51
Figure 2.21 CDT of a set of building objects with constraining edges is shown in solid lines while other virtual edges are shown in dashed lines.	52
Figure 4.1 (a) CDT with duplicate nodes of triangles hatched in grey colour and (b) Hashtable data structure to handle duplicates.	80
Figure 4.2 (a) CDT with a triangle comprising of duplicate nodes hatched in red colour and (b) the same hatched triangle with duplicate node IDNs generated from building number attached at three corners.	81
Figure 4.3 (a) Outer polygon and inner building polygons (triangulating features) with corner coordinates and (b) representation of the outer polygon and the two inner building polygons in ASCII format.	83
Figure 4.4 (a) Building polygons (triangulating features) with corner coordinates and (b) representation of two building polygons in ASCII format.	85
Figure 4.5 Constrained triangulation algorithm based on the polygon triangulation on ear-clipping by Eberly (2008).	90
Figure 4.6 (a) A simple data set of buildings and (b) a larger data set with buildings irregularly spaced, triangulated based on the polygon triangulation.	92
Figure 4.7 Building polygons with local IDNs in a single region surrounded by the road network.	93
Figure 4.8 Insertion point location: (a) inside triangle (b) outside convex hull of Δ_N , (c) on an edge of an interior triangle of Δ_N and (d) on an edge of a triangle bounded by convex hull to form the initial triangulation Δ'_{N+1}	95
Figure 4.9 Swapping procedure when inserting a point p into Delaunay triangulation.	96
Figure 4.10 Default Delaunay triangulation.	97
Figure 4.11 Representation of Delaunay triangulation.	98

Figure 4.12 Re-triangulation steps of the isolated polygons after subtraction of triangles and building polygons from the convex hull of all the site points P	99
Figure 4.13: Crossing triangles to be removed in the triangulation process.	100
Figure 4.14 (a) Constrained triangulation with duplicate nodes of a triangle hatched in red colour (b) same triangle with duplicate node IDNs and distances of the three edges D_1 , D_2 and D_3 and (c) array representing proximity links of both contiguous and disjoint buildings with the minimum Euclidean distance.	103
Figure 5.1 (a) Hierarchical structure of the road network (part) and (b) manual process of building grouping.	112
Figure 5.2 Hierarchical relationship between the three local constraints for building grouping. ...	113
Figure 5.3 Existing measures of building orientation.	114
Figure 5.4 Smallest minimum bounding rectangle (SMBR).	115
Figure 5.5 Calculation of the Hausdorff distance.	119
Figure 5.6 (a) DCT (b) adjacency relationship list (part) comprising of [bid_from, bid_to, proximity, orientation difference, similarity difference] and (c) MST segmentation in thick black lines with proximity as the weight.	121
Figure 5.7 (a) 2D initial adjacency matrix based on the proximity hierarchy depending on the target map scale and (b) example of the distribution of buildings spaced at three levels - VC, M and VF.	122
Figure 5.8 An example of linked pairs and non-linked pairs with building IDNs in a column of the adjacency matrix.	123
Figure 5.9 (a) Separation threshold between two buildings and (b) angle θ subtended by an arc of 0.125mm with radius $r = 1m$	128
Figure 5.10 Automatic building clustering. Source data at the scale of 1 : 1.25K and the clusters are formed from the source data at the target map scale of 1 : 5K for the subsequent map generalization.	130
Figure 5.11 Automatic building clustering on the source data at the scale of 1 : 1K.	131
Figure 5.12 Automatic clustering results in: (a) region 1 (b) regions 14, 15 and 16 (c) regions 11, 12 and 13 and (d) region 19.	136

Figure 5.13 (a) Partitioned regions surrounded by the road network where inner roads are ignored and (b) building features within each region.	138
Figure 5.14 Automatic building clusters (yellow) in region 9 used in the clustering experiment.	147
Figure 5.15 Generation of the CNDT using buildings and roads with the enforcement of their edges as constraints within region 9 used in the clustering experiment.	148
Figure 5.16 Automatic building clusters (light green) in region 9 after deriving adjacency relationships between buildings, taking into account the contextual inner roads within the region.	149
Figure 5.17 Building clusters on synthetic data.	151
Figure 5.18 Concave hull generation of a concave cluster of buildings on synthetic data.	152
Figure 5.19 Concave hull generation of a convex cluster of buildings on synthetic data.	153
Figure 5.20 Shape enrichment of clusters in three regions 4, 5 and 7 (part of) surrounded by the road network.	155
Figure 5.21 Shape enrichment of clusters in the same three regions 4, 5 and 7 (part of) surrounded by the road network as shown in Figure 5.20 with some improved results.	158
Figure 5.22 Data enrichment workflow of creating clusters and their subsequent shape enrichment.	162
Figure 6.1 Building at a corner.	171
Figure 6.2 Buildings with three neighbouring road segments around.	171
Figure 6.3 Different orientation of the LOBR of a building to the closest road.	173
Figure 6.4 LOBR of the building with two line segments connecting the centroid and midpoints of its edges.	174
Figure 6.5 Results of building orientation to the road on the OS MasterMap Data within a region surrounded by the roads.	177
Figure 6.6 Synthetic data set with its attributes at a decision point.	182
Figure 6.7 COBWEB clustering tree view (left) and the matrix showing cluster numbers against instances (right) of the synthetic data set in Figure 6.6.	184

Figure 6.8 Graphical representation of an enriched tree view with attributes, their values and instance IDNs (class IDNs) after the analysis.	184
Figure 6.9 Transformation (discretization) of the attribute - neighbour - in the synthetic data set in the WEKA GUI.	186
Figure 6.10 Classification output of the synthetic data set: (a) output at first iteration with the first instance hypothesized to be a landmark and (b) output at third iteration with the third instance hypothesized to be a landmark.	187
Figure 6.11 Classification output of the synthetic data set at the third iteration with the third instance hypothesized to be a landmark.	189
Figure 7.1 Generating group oriented bounding rectangle (GOBR).	200
Figure 7.2 Building enlargement: (a) enlargement along the Y-axis (height) and (b) enlargement along the X-axis (width) to comply building edges with the minimum building length.	201
Figure 7.3 Example illustration of a buffer around a polygon with cap square and join style mitre.	203
Figure 7.4 A pair of buildings in a cluster at different positions: (a) almost overhanging, (b) corner touching and (c) total separation.	203
Figure 7.5 Polygon geometry shown in red colour using the dilation operation and then the erosion with iterative shrinking for exceptional cases.	204
Figure 7.6 Squaring of an initial amalgam of a building cluster created using a synthetic data set with ear polygons.	206
Figure 7.7 Creating rectangular strips required in the enlargement process.	211
Figure 7.8 Selection of filling rectangular strips in the enlargement process.	212
Figure 7.9 Possible cases of building enlargement with narrow sections.	213
Figure 7.10 Introduction of granular edges in the amalgam after squaring and enlargement.	214
Figure 7.11 Simplification of the amalgam with orthogonal sides.	216
Figure 7.12 Functional process of building symbolization and aggregation with orthogonal sides.	217

Figure 7.13 Result of buffering with same positive and negative distance: (a) source cluster and (b) buffered amalgam with the two leftmost buildings (IDNs 1 and 2) bridged together, and two other source buildings (IDNs 3 and 4) in the cluster.	223
Figure 7.14 DCT on the buffered amalgam.	224
Figure 7.15 Triangles selected for bridging based on the distance threshold highlighted in yellow colour.	225
Figure 7.16 Results of the aggregation of building clusters with orthogonal sides on a real data set representing a region after squaring, enlargement and simplification.	228
Figure 7.17 Result of buffering with the same positive and negative distance: (a) source cluster with the lower building comprising of two concave corners shown circled and (b) buffered amalgam where the two concave corners are filled as a result of the buffering operation.	231
Figure 7.18 Result of buffering with the same positive and negative distance: (a) source cluster including a polygon with a hole and (b) buffered amalgam in a multi-polygon with a filling bridge between the two leftmost buildings and the preserved hole of the rightmost building in the source cluster in (a).	231
Figure 7.19 DCT on the buffered amalgam.	232
Figure 7.20 Possible cases of bridging a pair of buildings.	234
Figure 7.21 Dealing with polygon holes after aggregation of buildings with non -orthogonal sides.	237
Figure 7.22 Functional process of building aggregation with non-orthogonal sides.	238
Figure 7.23 Results of the aggregation of non-orthogonal shaped clusters comprising of buildings with non-orthogonal sides and/or significant orientation difference.	241
Figure 7.24 Results of the generalized amalgams with the improved concave hull algorithm.	242
Figure 8.1 Results of the CNDT with the edge constraints.	247
Figure 8.2 Results of clustering: (a) incorrect clustering due to topologically incorrect triangulation (highlighted in yellow colour) and (b) correct clustering after the fix of CNDT with the edge constraints.	248
Figure 8.3 Results of building orientation to the closest road.	249
Figure 8.4 Results of building orientation to the closest road after fixing the exceptions.	250

Figure 8.5 Results of building orientation to the closest road, located at corners.	251
Figure 8.6 Enlargement results after squaring with a distance 5m.	251
Figure 8.7 Simplification of a building with the OpenCarto edge deletion algorithm.	254
Figure 8.8 Location map of the London Boroughs of Newham and Tower Hamlets (hatched). ...	256
Figure 8.9 Newham test area: (a) road network with the classification of roads comprising of minor roads, private roads and alleys and (b) chosen regions (area partitions) for data processing during the focus map generation.	257
Figure 8.10 Source map of Newham test area at the scale of 1 : 8K.	258
Figure 8.11 Generalized building amalgams at the target scale of 1 : 8K to be used as the background on the focus map, derived from the source data at the scale of 1 : 1.25K in Newham area (part of).	259
Figure 8.12 Focus map with salient building landmarks highlighted with the graphical variable - colour - portrayed in the original shape on the coarse background of the generalized buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Newham area (part of).	260
Figure 8.13 Tower Hamlets test area: (a) road network with the classification of roads comprising of minor roads, private roads and alleys and (b) chosen regions (area partitions) for data processing during the focus map generation.	261
Figure 8.14 Source map of Tower Hamlets test area at the scale of 1 : 8K.	262
Figure 8.15 Generalized building amalgams at the target scale of 1 : 8K to be used as background on the focus map, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of).	263
Figure 8.16 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized amalgams of buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of).	264
Figure 8.17 Generalization results of clusters of orthogonal shape: (a) three source clusters highlighted in yellow colour (b) generalized amalgams with a distance of 4m (c) generalized amalgams in the ArcGIS software with a distance of 4m and (d) generalized amalgams in the ArcGIS software with a distance of 16m.	268

Figure 8.18 Generalization results of clusters of non-orthogonal shape with the research tool..	269
Figure 8.19 Generalization results of clusters of non-orthogonal shape with the ArcGIS software.	270
Figure 8.20 Comparison of amalgams of the non-orthogonal shaped cluster depicted in Figure 8.19(a) above.	271
Figure 8.21 Generalization results of clusters of orthogonal shape: (a) two source clusters highlighted in yellow colour (b) amalgams with an aggregation, exaggeration and simplification distance of 4m with the research tool (c) amalgams in (b) overlaid with source clusters in (a), and (d) amalgams after simplification with a tolerance of 4m using the ArcGIS software.	272
Figure 8.22 Generalization results of a cluster of non-orthogonal shape: (a) source cluster highlighted in yellow colour (b) amalgam with an aggregation distance of 4m and a space triangle edge threshold of 4m (c) amalgam before simplification with the two thresholds with a value of 8m (d) amalgam after simplification with 4m tolerance with the research tool (e) amalgam before simplification with a distance of 8m and (f) amalgam after simplification with a tolerance of 4m with the ArcGIS software.	273
Figure 8.23 Generalization results of a cluster of non-orthogonal shape: (a) two source clusters highlighted in yellow colour (b) amalgams with an aggregation distance of 4m and a space triangle edge threshold of 4m with the research tool and (c) amalgams with an aggregation distance of 4m with the ArcGIS software, no simplification applied in both cases.	275
Figure 8.24 (a) Salient landmarks (highlighted in yellow colour) derived from region 3 of Newham area (see Figure 8.9), delineated in blue and (b) salient landmarks (highlighted in yellow colour) derived in sub-regions 3A and 3B of the main region 3, applying the J48 implementation in each sub-region separately.	277
Figure 8.25 Landmark saliency results (highlighted in yellow colour) of region 5 of Newham area: (a) results by the J48 implementation and (b) results by the framework of Raubal and Winter (2002).	284
Figure 8.26 Landmark saliency results (highlighted in yellow colour) of region 1 of Tower Hamlets area: (a) results by the J48 implementation and (b) results by the framework of Raubal and Winter (2002).	284

Figure 8.27 Landmark saliency results (highlighted in yellow colour) of region 5 of Newham area: (a) results by the J48 implementation and (b) results by the new method on the MAD developed in this research.	290
Figure 8.28 Landmark saliency results (highlighted in yellow colour) of region 1 of Tower Hamlets area: (a) results by the J48 implementation and (b) results by the new method on the MAD developed in this research.	291
Figure 8.29 Focus map of Newham area with the insets A, B, C and D of the specific locations considered in the validation of the landmark saliency.	292
Figure 8.30 Salient landmark visualization on Swete Street.	293
Figure 8.31 Salient landmark visualization on Ballam Street and Dongola Road West.	294
Figure 8.32 Salient landmark visualization on Whitwell Road and Grant Street.	295
Figure 8.33 Salient landmark visualization on Barking Road.	296
Figure 8.34 Focus map of Tower Hamlets area with the insets A, B and C of the specific locations considered in the validation of the landmark saliency.	300
Figure 8.35 Salient landmark visualization on Goldsmith's Row.	301
Figure 8.36 Salient landmark visualisation on St. Peter's Close.	302
Figure 8.37 Salient landmark visualisation on Centre Street.	303
Figure 8.38 Google street view of high-rise buildings with the same shape and the height visualized from the middle of the Centre Street.	306
Figure 8.39 Focus map of Tower Hamlets area with the decision point circled, and the regions (1 to 5) used in deriving landmark saliency.	307
Figure 8.40 Evaluation of salient landmarks at a decision point with the J48 implementation: (a) Salient landmarks falling within the buffer of 50m, derived from each region separately (regions 2, 3, 4 and 5 in Figure 8.39 and (b) salient landmarks derived from all the buildings within the same buffer at the decision point during focus map generation in Tower Hamlets area.	308
Figure 8.41 Evaluation of the landmark saliency at a decision point in Tower Hamlets area: (a) all the buildings around the decision point within a radius of 50m (b) salient landmarks chosen by the J48 implementation and (c) salient landmarks chosen by the implementation of the framework of Raubal and Winter (2002) from the buildings with a total significance ≥ 0.75	309

Figure 8.42 Evaluation of the landmark saliency at a decision point in Tower Hamlets area with all the buildings around the decision point within a radius of 50m: (a) salient landmarks chosen by the framework of Raubal and Winter (2002) from the buildings with a total significance ≥ 0.75 and (b) salient landmarks chosen by the new method on the MAD from the buildings with a total significance ≥ 1 and the significance contribution from two or more significance measures.	312
Figure 8.43 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Newham area (part of).	321
Figure 8.44 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized amalgams of buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of).	322
Figure A.1 (a) Dialogue menu with the coordinate values of the MBB of the building data set retrieved automatically for generating outer polygon of the input data structure and (b) input building geometries in ASCII format where polygon IDN of the outer polygon (MBB) is assigned -1.	350
Figure A.2 (a) Dialogue menu to extract and save building polygon geometries in ASCII format and (b) extracted building outer polygon geometries with respective building IDNs.	351
Figure B.1 (a) CDT output with rectangular buildings outlined in red colour and their IDNs on the GUI and (b) adjacency links between buildings where Polygon IDN 1 is the IDN of the outer polygon.	352
Figure B.2 (a) CNDT output and (b) output on polygon triangulation based algorithm with the edges of building polygons set as constraints, both implemented on the same GUI.	353
Figure B.3 GUI for spatial clustering of building polygon geometries in the data enrichment process with the use of constrained algorithm on Delaunay triangulation.	354
Figure B.4 Distance weighted initial MST in thick black lines used for clustering of building geometries.	355
Figure B.5 UI for matching automatic clustering results with that of manual clustering by the subjects in both expert and lay groups.	356

Figure B.6 GUI for spatial clustering of building polygon geometries with the consideration of the contextual features (roads in black colour) using the CNDT with edge constraints.	357
Figure B.7 Cluster shape enrichment of building clusters.	358
Figure B.8 Generalization GUI for building geometries.	359
Figure C.1 Contribution of an edge of a building to a candidate orientation α_i on modulo $\pi/2$	362
Figure C.2 Simplification of shorter edges.	363
Figure D.1 Source map at 1 : 4K (left) and the target map reduced and printed at 1 : 10K (right).	364
Figure D.2 Regions surrounded by the road network.	365
Figure D.3 Building features in each region depicted by a unique colour.	366
Figure D.4 Manual clustering output of a subject from the expert group.	368
Figure D.5 Manual clustering output of a subject from the lay group.	369
Figure F.1 Sensitivity analysis workflow in the WEKA GUI.	400
Figure F.2 Output of the sensitivity analysis in the WEKA GUI.	401
Figure F.3 Attribute transformation with the unsupervised discretization.	402
Figure F.4 Data mining UI for extracting the salient landmarks using the algorithms - CobWeb, ID3 and J48 - implemented with the open source WEKA Java APIs.	403
Figure F.5 User interface for the salient landmark evaluation based on the two frameworks of (a) Raubal and Winter (2002) and (b) Nothegger, Winter and Raubal (2004), and the method on the MAD developed in this research.	405

List of Tables

Table 4.1 Results of the CDT based on the sweep line algorithm by Domiter and Žalik (2008) with different data sets.	84
Table 4.2 Results of the CNDT based on the incremental algorithm by Ruppert (1995).	88
Table 4.3 Results of the constrained triangulation developed in this research based on the polygon triangulation algorithm by Eberly (2008) on the source data.	91
Table 4.4 Results of the constrained triangulation developed in this research based on the Delaunay triangulation with the recursive edge-flipping technique (Berg <i>et al.</i> , 2008) using the incremental method.	101
Table 4.5 Computation times to generate triangulation and neighbourhood relations.	104
Table 4.6 Validation of the constrained triangulation developed using the edge deletion method (Shewchuk, 1999).	105
Table 4.7 Different triangulation types implemented and compared.	108
Table 5.1 Typical examples of the orientation difference between a rectangular pair of buildings based on the wall orientation algorithm by Duchêne <i>et al.</i> (2003).	116
Table 5.2 Main polygon building clustering algorithm with the contributing algorithms.	132
Table 5.3 Summary of the clustering results of the expert group derived from the cluster data given in Appendix D.6 in each of the twenty two regions compared with automatic clustering results.	134
Table 5.4 Summary of the clustering results of the lay group derived from the cluster data given in Appendix D.6 in each of the twenty two regions compared with automatic clustering results.	135
Table 5.5 Summary of the results in identifying the medium distance range clusters for the hierarchical application of the Gestalt constraints - orientation and similarity difference - by the expert and the lay groups, derived from the cluster data given in Appendix D.6.	137
Table 5.6 Summary of the results of percentages of the subjects in both groups identifying clusters with a misclassification, derived from the cluster data given in Appendix D.6.	139
Table 5.7 Evaluation of the automatic clustering method by the expert and the lay groups.	140

Table 5.8 Evaluation of the use of threshold values of the Gestalt factors - proximity, orientation and similarity in the automatic clustering method by the expert and the lay groups.	141
Table 5.9 Evaluation of the comparison of manual clustering approach with the automatic clustering by the expert and the lay groups.	142
Table 5.10 Evaluation of the adaptation of manual clustering process by the expert and the lay groups.	143
Table 5.11 Summary of the answers to the Question 3(b) of the questionnaire in Appendix D.7	143
Table 5.12 Summary of the answers to the Question 4(b) of the questionnaire in Appendix D.7	144
Table 5.13 Summary of the answers to the Question 6(b) of the questionnaire in Appendix D.7	145
Table 5.14 Summary of the answers to the Question 9(b) of the questionnaire in Appendix D.7	145
Table 5.15 Enlarged view of the clusters delineated in Figures 5.20 and 5.21 for the clear view of the orientation difference.	159
Table 5.16 Cluster shape enrichment algorithm with the contributing algorithms.	161
Table 6.1 Description of attributes considered to derive salient landmarks in the data mining process.	166
Table 6.2 Building categories and their priority rankings.	179
Table 6.3 Instances with attributes chosen after the sensitivity analysis on the synthetic data set where the attribute lmark is the classifier used as a dependent variable.	187
Table 6.4 Results of the COBWEB clustering on the top level of the tree.	192
Table 6.5 Results of the COBWEB clustering using the top two levels of the tree.	192
Table 6.6 Results of the ID3 classification.	193
Table 6.7 Results of the J48 classification on the top level of the tree.	194
Table 6.8 Results of the J48 classification using the top two levels of the tree.	194

Table 6.9 Comparison of the results of the three data mining algorithms using the top level and the very next lower level (1 st and 2 nd physical levels) of the output trees on a building data set spread over a region enclosed by roads.	197
Table 7.1 Test results of the building aggregation algorithm with dilation and erosion followed by squaring the sides of the amalgam.	208
Table 7.2 Test results of the building aggregation algorithm with concave hull generation followed by squaring the sides of the amalgam.	209
Table 7.3 Test results I of building aggregation algorithm with orthogonal sides.	218
Table 7.4 Test results II of the building aggregation algorithm with orthogonal sides.	220
Table 7.5 Results of further testing of the building aggregation algorithm with orthogonal sides using an exceptional synthetic data set.	221
Table 7.6 Results of the aggregation of building clusters with orthogonal sides on the synthetic data used in Table 7.5 with some exceptional building configuration.	227
Table 7.7 Building cluster aggregation algorithm with orthogonal sides, including contributing algorithms.	229
Table 7.8 Pairing algorithm of adjacent triangles.	236
Table 7.9 Results of aggregation of building clusters with non-orthogonal sides on the synthetic data with some exceptional building configuration.	240
Table 7.10 Building cluster aggregation algorithm with non-orthogonal sides, including contributing algorithms.	243
Table 8.1 Evaluation of the generalization constraints on amalgams of Newham data.	271
Table 8.2 Evaluation of the generalization constraints on amalgams of Tower Hamlets data.	276
Table 8.3: Method of calculating the overall significance score of each building using individual significance value of each attribute based on the framework by Nothegger, Winter and Raubal (2004).	279
Table 8.4 Deriving a total significance score for a particular building by the method of Raubal and Winter (2002).	281
Table 8.5 Evaluation of landmark saliency of the focus map with the framework of Raubal and Winter (2002) in the regions depicted in Figure 8.9, page 257 of Newham area.	282

Table 8.6 Evaluation of landmark saliency of the focus map with the framework of Raubal and Winter (2002) in the regions depicted in Figure 8.13, page 261 of Tower Hamlets area.	282
Table 8.7 Attributes and their transformed values to be compatible with the new landmark saliency measure.	287
Table 8.8 Deriving a total significance measure for landmark saliency for a building to be applied to the new method based on the MAD.	288
Table 8.9 Evaluation of landmark saliency with the new method developed on the MAD in the Newham data set.	289
Table 8.10 Evaluation of landmark saliency with the new method developed on the MAD in Tower Hamlets data set.	289
Table 8.11 Landmark significance together with attributes and their values of the four chosen salient landmarks with the J48 implementation.	310
Table 8.12 Landmark significance together with attributes and their values of the four chosen salient landmarks with a total significance ≥ 0.75 based on the framework by Raubal and Winter (2002).	310
Table 8.13 Comparison of landmark significance together with individual significance scores of each measure of the most prominent salient landmarks chosen based on the framework by Raubal and Winter (2002) and the new method based on the MAD.	312
Table D.1 Results of the expert group in each partitioned region.	370
Table D.2 Results of the lay group in each partitioned region.	381
Table F.1 Enriched test data (part of) of a region surrounded by the road network. Data source: OS MasterMap.	404
Table F.2 Results of the salient landmarks at a decision point.	406

List of Abbreviations

AGENT: Automatic GENaralization New Technology

API: Application Programming Interface

CDT: Constrained Delaunay Triangulation

CNDT: Conforming Delaunay Triangulation

DCM: Digital Cartographic Model

DCT: Delaunay Constrained Triangulation

DEM: Digital Elevation Model

DF: Distance Factor

DLM: Digital Landscape Model

DHD: Discrete Hausdorff Distance

ESRI: Environmental System Research Institute

GIS: Geographic Information System

GPS: Global Positioning System

GSP: Good Sub-Polygon

GOBR: Group Oriented Bounding Rectangle

GUI: Graphical User Interface

HD: Hausdorff Distance

ICA: International Cartographic Association

IDN: Identification Number

JCS: Java Conflation Suite

JTS: Java Topology Suite

KDD: Knowledge Discovery in Database

LBS: Location Based Services

LOBR: Locally Oriented Bounding Rectangle

MAD: Median Absolute Deviation

MBB: Minimum Bounding Box

MBR: Minimum Bounding Rectangle

MRDB: Multiple Representation Database

MRDBMS: Multiple Representation Database Management System

MSL: Mean Sea Level

MST: Minimum Spanning Tree

NMA: National Mapping Authority

OGC: Open Geospatial Consortium

OS: Ordnance Survey

POI: Points of Interest

PSLG: Planar Straight Line Graph

RNG: Relative Neighbourhood Graph

SMBR: Smallest Minimum Bounding Rectangle

STR: Sort-Tile-Recursive

2D: Two-Dimensional

3D: Three-Dimensional

UI: User Interface

Acknowledgements

First and foremost, I want to express my deepest and sincere gratitude to my main supervisor and the director of studies Professor Allan J. Brimicombe for his continuing guidance, advice and encouragements throughout the duration of my study. I had the opportunity to learn a lot in the field of research through his vast experience as a cross-disciplinary researcher. I would like to thank Dr. Yang Li, my second supervisor for his constructive criticisms, suggestions and valuable feedback that contributed to the successful completion of the research.

I take this opportunity to offer my special appreciation to all those who provided valuable support in sharing their knowledge and ideas to solve the issues I encountered in my work through the open source user forums. Special thanks should go to Dr. Martin Davis - the developer of the Java topology suite (JTS): a two-dimensional (2D) spatial predicates library, Mr. Micheal Bedward – the geographical information system developer of OpenJump and Dr. Stefan Steinger - Postdoctoral Research Associate from the University of Calgary and Pontifica Universidad Católica de Chile - for providing me with open source libraries and useful resources in the area of the research.

I extend my gratitude to the school of Architecture, Computing and Engineering, and the Graduate school for the support provided to carry out the research, ensuring an excellent atmosphere from the beginning to the end.

My sincere thanks should also be extended to the Ordnance Survey of the United Kingdom for the production of highly detailed Ordnance Survey MasterMap data that I used in my work for conducting the research.

This acknowledgement would not have been complete without mentioning the name of the National Mapping Authority in Sri Lanka for setting an encouraging platform to peruse higher studies and providing useful resources for conducting the research.

Dedication

This dissertation is dedicated to my loving family for the constant support and encouragement that has made this work possible. Finally, I dedicate this work in memory of my father who always encouraged me to the pursuit of academic excellence.

Chapter 1 Introduction

This chapter provides the motivation for taking up this research where it discusses the background of wayfinding and broad aspirations of what is to be achieved through this research.

1.1 Research motivation

The use of geographic information through mobile applications has been rapidly increasing and enabling users to view their current position in the context of the neighbouring environment through the advancement of technologies such as Global Positioning System (GPS) and ubiquitous computing Internet technologies, along with the availability of mobile devices. Thus, one of the major challenges facing the national mapping authorities (NMAs) across the world is the development of maps that are more helpful and attractive to be used in mobile applications such as in wayfinding. This necessity is emphasised in the usability evaluation of topographic maps for mobile devices where users need more meaningful map entities in topographic maps that should be adapted, according to their context of use according to Nivala *et al.* (2003), since cartographic presentation and symbology in traditional topographic maps are not designed for wayfinding applications.

Wayfinding is the process by which human beings orient themselves and navigate through space. According to Allen (1999) wayfinding by human beings can be mainly for three purposes: (a) travel with a goal of reaching a familiar destination (b) exploratory travel with the view of returning to a familiar point of origin and (c) travel with the goal of reaching a novel destination. Therefore, it is evident that wayfinding involves direct interaction between the traveller and the environment. Human beings use various spatial, cognitive and behavioural abilities to find their way through the environment, gaining environmental information or spatial knowledge about the environment (Raubal and Winter, 2002; Lloyd, 1989; Gopal and Smith, 1990; Cornell, Sorenson and Mio, 2003; Brimicombe and Li, 2010). According to Kuipers (1978), people acquire spatial knowledge in positioning through route descriptions, topological relations of the road network and

the orientation of objects in the environment. Research in spatial cognition has shown that maps are a vital means of providing spatial knowledge to assist travellers acquire route information to reach their destination without trouble (MacEachren, 1995; Kray *et al.*, 2003; Elias and Paelke, 2008).

Further experiments conducted by Denis *et al.* (1999), Freksa *et al.* (1999), Tversky and Lee (1999) and Montello, Michon and Denis (2001) have shown that pedestrians perceive landmarks as a useful part of route information in wayfinding. Lynch (1960) describes landmarks as defined physical objects external to the observer and that are used as a point of reference to make one orient oneself. According to Golledge (1999), landmarks are physically defined objects that stand out from the surroundings and help locate geographic position. More meaningfully, landmarks are cognitively distinct from other elements in spatial memory and central to the nature and organisation of spatial representation (Presson and Montello, 1988).

Thus, including supplemental landmarks could make travellers much more confident and comfortable when experiencing a new environment irrespective of the navigation aids whether they be a traditional paper street map or a vehicle navigation system (Deakin, 1996). However, according to Deakin (1996), one of the reasons for the non-inclusion of supplemental landmarks on street maps arises from the difficulty of selecting such landmarks that are salient based on a standard methodology and criteria. Another issue is the limited map space to incorporate landmarks, which requires the application of map generalization techniques to reduce map details to accommodate space. Although current navigation systems utilise visual representation in addition to positioning and routing functionality to convey navigational information to the users, landmarks are still missing in spatial data sets in such applications due to aforesaid limitations. However, various approaches by Raubal and Winter (2002), Elias (2003), Elias and Brenner (2005) and Elias, Hampe and Sester (2005) have been attempted to derive salient landmarks from spatial data sets to improve navigation aids. Evaluation of such methods in terms of geometrical, spatial and semantic characteristics of such landmarks for the effective integration of them into both static maps and mobile maps (maps that can be accessed wirelessly to use

in mobile situations (Meng and Reichenbacher, 2005) for wayfinding) has not been investigated yet. Further, due to the technical limitations such as processing power, memory, resolution and small screen size in mobile devices, the general cartographic rules in applying colours, symbology and feature representation in map production could not be used in mobile mapping. To this effect, some research has discussed the cartographic repercussions of small displays (Gartner and Uhlig, 2001; Radoczky and Gartner, 2005; Elias, Hampe and Sester, 2005).

Therefore, it is understood that there is a significant difference between a traditional topographic map and a map designed for mobile navigation especially considering the limitations discussed above. Another important factor in designing a mobile map is to enable users to engage in the wayfinding task by presenting him/her an egocentric map view which is more a technique of representing geographic information in relation to a user's position (Meng, 2005). This not only keeps users focused on the task of interest without distracting him/her from other external interferences in the environment, but also improves his/her cognitive capability of understanding the immediate surroundings of the navigation route in a manner which is not the same as gathering spatial knowledge by reading a traditional topographic map. In a traditional topographic map, a uniform visual balance of details is maintained with an allocentric view (viewing location of one object in relation to other objects) throughout the map at a uniform scale. This emphasises that mobile applications for wayfinding should allow users to bring to their attention the areas of interest rather than searching for important locations and prominent features, reading through the whole map as is the case in retrieving information from a traditional analogue topographic map.

Thus, the NMAs should aim to produce customer-oriented mobile maps, incorporating important landmarks that are more useful and attractive to users in finding points of interest with the minimum effort using cartographic visualisation techniques discussed above rather than necessarily giving priority to producing conventional topographic map series along their production lines. Being a surveyor by profession in the NMA of Sri Lanka, having read Master of Science Degree in Geoinformatics at the Faculty of Geo-information

Science and Earth Observation (ITC) of the University of Twente, the Netherlands, with a specialisation in multiple representation visualisation of topographic features of the same phenomena, I have been persuaded to take up this research which is oriented towards investigating the automatic processes of developing visually appealing maps primarily designed for wayfinding users to bring their immediate focus to the area of interest with highly detailed, prominent features and leaving all other information in the peripheral areas with the coarse background information.

1.2 Background

With the growth of ubiquitous computing through the emergence of mobile devices and distributed applications over the Internet, personalisation is leaving the desktop domain (Günter, 1991; Zipf and Jöst, 2006). A widespread availability of mobile devices and the distributed applications via wireless access have allowed people to access maps in mobile situations personally rather than using fixed line, personal computer based web maps. Particularly, ubiquitous computing has brought location based services (LBS) into existence with a variety of spatial applications. LBS are regarded as services that deliver data and information customised to the current or some projected location and context of the user (Brimicombe and Li, 2006). As one of the principal and useful applications in LBS, using lightweight mobile devices to point the location on Earth has been popular with the rapid development of geographic information system (GIS), GPS, radio frequency identification and various other location sensing technologies with varying degrees of accuracy (Jiang and Yao, 2006). According to Reichenbacher (2003), a mobile user can perform various tasks in relation to geoinformation in the real world: locating, navigating, searching, identifying and checking. Maps containing these geoinformation can either be stored in the mobile device or retrieved online. *Locating* is related to the question “where am I?” or “where is X?”. *Navigating* involves wayfinding either from the user’s current position to a given position or object or between any specified objects/positions independent of the user’s current location. *Searching* deals with finding objects or people located from the user’s current position. *Identifying* relates to the recognition of objects in relation to the other objects in the vicinity. Finally *checking* relates to an event task for

finding information about what happens at a given place. As a result of the capabilities of such spatially related tasks, wayfinding has become one of the most popular applications for small electronic devices such as smartphones. Further, in contrast to in-vehicle navigation systems (such as SatNavs), the increasing demand for the use of mobile phones has also brought pedestrian wayfinding to the fore (Elias, Hampe and Sester, 2005). Such pedestrian wayfinding systems depend heavily on maps to convey wayfinding information to the users in addition to positioning and routing functionality (Elias and Paelke, 2008).

However, different visualisation techniques such as changing the opacity/colours of features (Reichenbacher, 2004, 2005; Figure 1.1 below) and application of the variable scale object representation techniques (Fairbairn and Taylor, 1995) have been attempted to improve the egocentric map. In addition to the technique of visualising multiple representations of data at different scales (Elias, Hampe and Sester, 2005) based on map generalization, that is, the reduction of details of a source map at a larger scale to produce a target map at a smaller scale to accommodate smaller map space. Further, a mobile map should have a task-based representation, concentrating on rich details in the area of interest while representing coarse details in the peripheral area rather than an exploratory instrument like a traditional topographic map with uniformly rich detail to gain an idea of the entire area of the map (Meng, 2005). This emphasises that a mobile map should represent highly selective information in accordance with the purpose of use (user context and objectives). Due to specific needs and technical limitations of the wayfinding systems, a relatively new concept called adaptation has been introduced in mobile geographic information applications (Zipf and Jöst, 2006). The two most important factors for adaptation are (a) the user's objectives and (b) the context in the representation of the current situation of the immediate navigation environment. The aforesaid selective information in the immediate navigation environment should include salient landmarks as described in Section 1.1 for users to aid navigation. According to the study conducted by Lovelace, Hegarty and Montello (1999), most frequently used landmarks in finding route directions in unfamiliar environments are point landmarks (at decision points) and on-route landmarks (along the path). In addition, two other types of landmarks, distinguished by Lovelace, Hegarty and Montello (1999), are potential choice

point landmarks (street intersections) and off-route landmarks (distant but visible from the route). According to Deakin (1996), potential choice point landmarks are not distinct in an urban environment. Also, off-route landmarks are only used in navigation by a novice for overall guidance according to Lynch (1960). Further, it has been found that about 50% of all the landmarks in wayfinding instructions are buildings according to a study conducted by Elias and Paelke (2008). Therefore, a wayfinding application heavily relies on the identification of salient buildings at decision points (choice point) and along the path (on-route) of navigation.

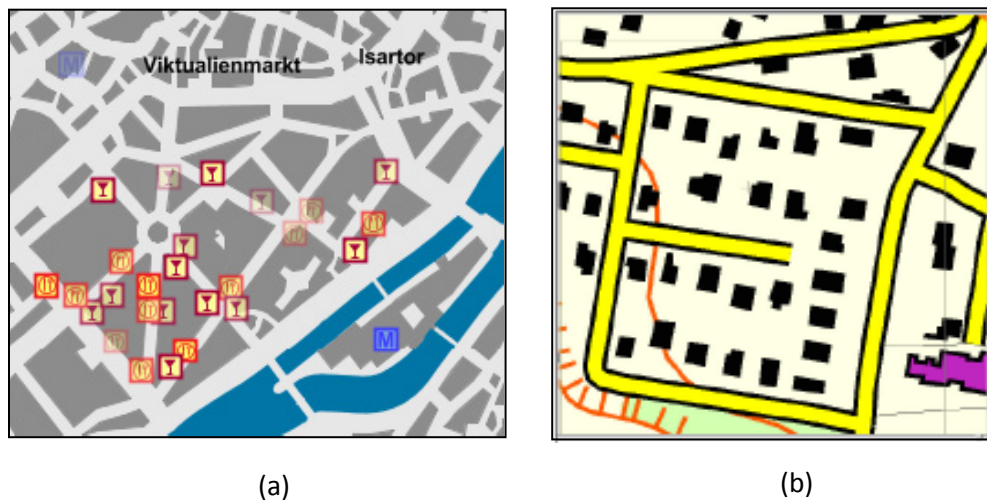


Figure 1.1 (a) Mobile map with symbols represented by relevance values with different opacities, from Reichenbacher (2005) and (b) traditional cartographic map, from Bard (2003).

From the definition of LBS according to Brimicombe and Li (2006), it is understood that the LBS mainly consist of a system (hardware and software), user, location and context (environment). These components play a significant role in LBS applications. A number of models of these components in LBS have been presented in the literature (Sarjakoski and Nivala, 2005; Jiang and Yao, 2006; Li, 2006). According to Sarjakoski and Nivala (2005), when a mobile map in LBS is used in the field, a user has to deal with different contexts in the surrounding environment depending on the type of application (Figure 1.2).

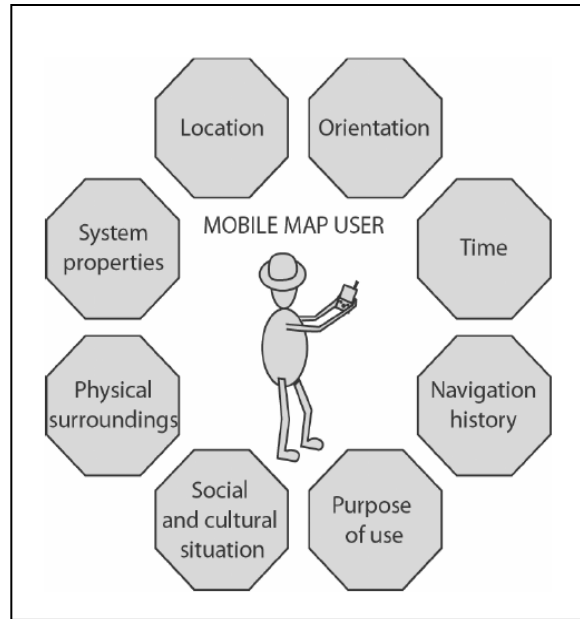


Figure 1.2 The surrounding context of a mobile map user, from Sarjakoski and Nivala (2005).

Location deals with the real time information about a wayfinder's current position on the mobile map which cannot be retrieved from a traditional hardcopy map. Wayfinding efficiency and effectiveness has a high dependency on system properties of the mobile device in terms of screen size, resolution, memory capacity and processing speed. The mobile map created should fit the situation and the purpose so that depending on the user requirement, different specific map views should be able to be generated (e.g. during daylight on a sunny day a mobile map with coloured landmarks is more useful than at night time). Also during wayfinding, depending on the time of the day, points of interest (POI) along the route may differ considerably. Another important factor in the context is the physical surroundings of the user. For example, screen display colour and brightness should adapt to the light of the physical surroundings irrespective of daylight or night and the environmental condition (a sunny day or rainy day). Further, the loss of orientation in an underground environment and disruptions, such as busyness of people during peak times, to navigation in an urban environment are some other impediments in physical surroundings. The observer's viewpoint is also an important factor to be considered

together with the route selection based on the topography of the area of navigation. It is also important to a user to retrieve the navigation history and the planned route in some situations. Orientation is considered where the mobile map should be displayed in the right position with respect to the user's line of sight and the direction of movement. Usage situations of mobile maps may also vary with a user's social and cultural settings (Sarjakoski and Nivala, 2005). In addition to the purpose of use, one of the most challenging contexts to handle is the characteristics of the user him/herself. The reason is that the physical, cognitive, perceptual abilities and the personality of the user will have a considerable effect on their wayfinding scenario. However, the main aspiration of this research is towards improving the context with more emphasis on the enrichment of salient landmarks and enhancement of visual representation by way of investigating the processes to generate a focus map view (see Figure 2.1 in Section 2.1 for a variable scale focus map) using multiple representations of spatial data. It will also help NMAs produce the topographical maps and the task-oriented wayfinding maps with the inclusion of salient landmarks using a methodical and systematic approach which is still lacking in commercial navigation data sets (Elias and Paelke, 2008) and current map production at NMAs.

1.3 Outline of the thesis

Following this introductory chapter, this thesis is organised into nine chapters, and they are briefly described as follows:

Chapter 2 describes the research context. It first introduces focus maps and then presents a review of the related work in each field involved in generating focus maps. Then the overall objective is defined and research questions are formulated.

Chapter 3 presents the methodology as to how it is proposed to achieve the overall objective through answering the research questions formulated in the previous chapter.

Chapter 4 describes the testing and evaluation of existing algorithms so as to develop and implement a new constrained triangulation spatial data structure to derive explicit neighbourhood relations between polygon geometries.

Chapter 5 describes how two new data enrichment algorithms: (a) spatial clustering of building polygons and (b) shape enrichment of such clusters are developed and implemented through testing, evaluating and modifying existing algorithms to be used in subsequent automatic map generalization.

Chapter 6 describes how new algorithms and methods are developed to enrich attributes that are required to derive landmark saliency of building features stored in a spatial database in the first part. In the second part, it further describes how the existing decision tree algorithms used in data mining are tested, modified and evaluated to derive the salient building landmarks with the enriched attributes during the first phase.

Chapter 7 describes how four different aggregation algorithms based on cluster characteristics enriched during the data enrichment process, are developed and implemented through testing, evaluating and modifying existing algorithms to be used in the automatic generalization of building polygons to generate a coarse background in focus maps.

Chapter 8 illustrates the results of focus map generation together with their external validation in the areas of automatic map generalization and data mining for deriving landmark saliency using the two test data sets chosen from Newham area and Tower Hamlets area of London, United Kingdom. Then it critically discusses what has been done in each area involved in generating focus maps in relation to the related work.

Chapter 9 is the conclusions where the results and the answers to the research questions are reaffirmed. Then it describes the significance of the work done in this research. Finally, it provides an outlook of the areas for future research.

Chapter 2 Research context

This chapter first introduces focus maps and then presents a critical review of related work in each area involved in generating focus maps for pedestrian wayfinding, which provides the basis for identifying the problem scope to formulate research questions. In a focus map, the user's attention is drawn to a specific area or location with cartographic representation techniques by way of emphasising salient features. In this research on generate focus maps, four specific areas are taken into account: Delaunay triangulation data structure, data enrichment, automatic map generalization and data mining under knowledge discovery in spatial databases. These areas serve as a basis to design, implement, test and evaluate methods for generating focus maps. Delaunay triangulation will be used to get the adjacency relationships between polygonal spatial features. Data enrichment is for extracting hidden information from the spatial databases to be utilised in subsequent automatic map generalization and data mining processes. Next the main technique used to generate focus maps is the automatic map generalization. Finally, in order to retrieve salient landmarks to be incorporated into focus maps, data mining techniques are used in the knowledge discovery process.

2.1 Focus maps

Fairbairn and Taylor (1995) have discussed methods to derive variable scale maps that are another form of focus maps based on a space-directed transformation technique (Bereuter and Weibel, 2010) in which map objects are moved apart to reduce conflicts by transforming the map space. In this process, the scale constantly decreases from the centre towards the edge of the map (Figure 2.1(a)). Further work on generating variable-scale maps has been carried out by Harrie, Sarjakoski and Letho (2002). In their approach the scale is kept constant within a circular cap based on the distance from the centre of the map while the scale of features outside the circular cap constantly decreases towards the edge of the map (Figure 2.1(b)). In both approaches, details have been reduced to avoid compressed data at the edge of the maps based on map generalization before applying variable scales. Fairbairn and Taylor (1995) have applied the variable scale

approach on two data sets where data at the centre represent large scale data while the data at the edge of the map represent small scale data. Harrie, Sarjakoski and Letho (2002) have chosen a large scale data set at the centre within the circular cap while data outside the circular cap are generalized with selection and simplification of details followed by applying a variable scale technique.

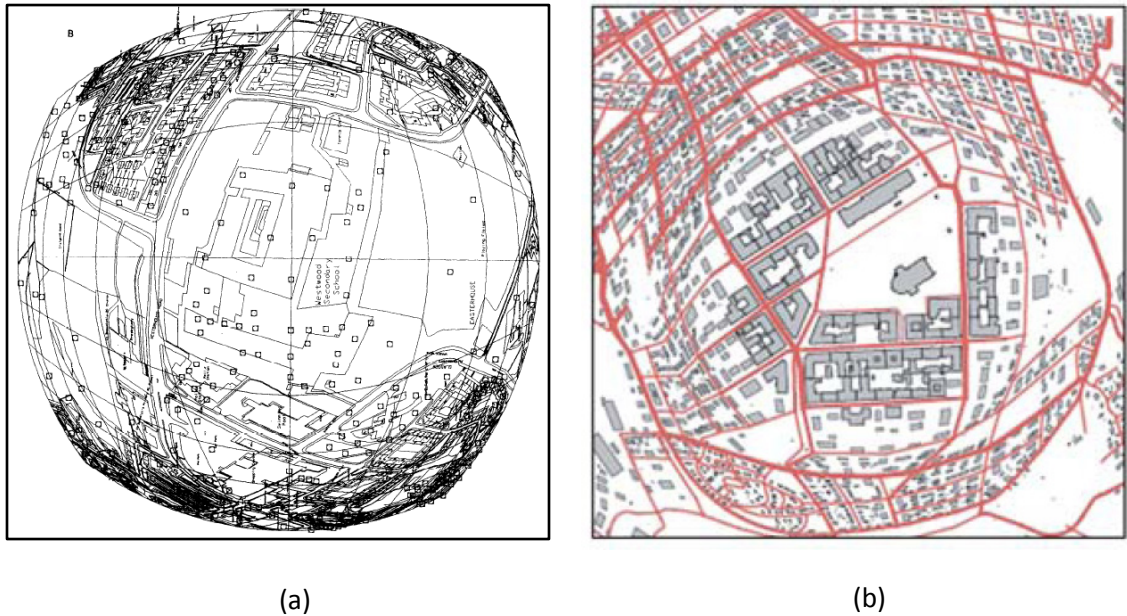


Figure 2.1 Focus maps represented with variable scales: (a) scale radially decreases constantly from the centre of the map, from Fairbairn and Taylor (1995) and (b) scale within a circular cap based on a distance from the centre is uniform while the scale outside the area of circular cap radially constantly decreases, from Harrie, Sarjakoski and Letho (2002).

A major characteristic of a variable scale map in wayfinding is to emphasise data in large scale at the region of interest to the user while the data in the peripheral area are represented at decreasing scales as distances increase from the centre of the area of interest. However, since the scale throughout the map is not uniform, features get cluttered towards the edge of the map even if the map generalization technique is applied to reduce details. Thus, the application of variable scale does not necessarily make the map more legible. Furthermore, the variable scale map is not visually attractive and lacks spatial fidelity because of the distortions. The evaluation of perceptual and cognitive validity of the variable scale approach in mobile map representation has not been tested.

To overcome the issues in distortions on variable scale maps, Rappo in 2003 as cited by Reichenbacher (2004), has used a new technique of generalization called radial generalization where the details of the map are radially generalized from the user's position towards the edge of the map.

Zipf and Richter (2002) have applied the focus map technique to emphasise the region of interest of the user according to their current task of wayfinding by enhancing the visualisation of landmarks by the assignment of colours and an object-directed transformation technique (Bereuter and Weibel, 2010). When the object-directed transformation is applied, it is the objects in the map that are modified using map generalization operations without changing the metric properties of the underlying map space. The same concept has been extended by Neis and Zipf (2008) to represent routes with 3D landmarks (buildings) using Open Geospatial Consortium (OGC) web service standards. Reichenbacher (2005), citing his own work on the egocentric representation of mobile maps (Reichenbacher, 2004), has described the concept he used to model the relevance of geographic objects in terms of opacity values by determining the relevance of events for mobile users, calculating the temporal and spatial distances to the events based on the current location and time.

Elias, Hampe and Sester (2005) have used an adaptive visualisation technique which is a focus map technique for presenting important building information to the user in real-time based on the multiple representation data handling as described by Hampe, Anders and Sester (2003), together with object-directed transformation of objects (map generalization). In this method all the building information is represented in the background with coarse details and as the user gets closer to a salient building landmark on the navigation route, the respective generalized coarse detail (building amalgam) is replaced by the salient landmark represented at the original scale. This replacement is achieved by the object links between the salient landmarks and the generalized building amalgams stored in a multiple representation database management system (MRDBMS). The salient landmarks are emphasised by using a visual variable - colour - as used by Zipf and Richter (2002) and Reichenbacher (2004). The adaptive visualisation mentioned

herein implies the use of the visual variable colour applied to the single objects of salient landmarks where the colour should be changed (adapted) depending on the environmental context of navigation.

2.2 Map generalization

Map generalization is the process of applying modifications to the information on a cartographic map so that information can be shown on a smaller surface area whilst retaining the essence of original geometrical and descriptive characteristics. According to the International Cartographic Association (ICA) in 1973, generalization is “the selection and simplified representation of detail appropriate to the scale and/or the purpose of the map” as cited by Brassel and Weibel (1988). This emphasises that the smaller the map scale, the more the representation is to be simplified and abstracted in order to maintain visual clarity and balance of the representation (Figure 2.2).

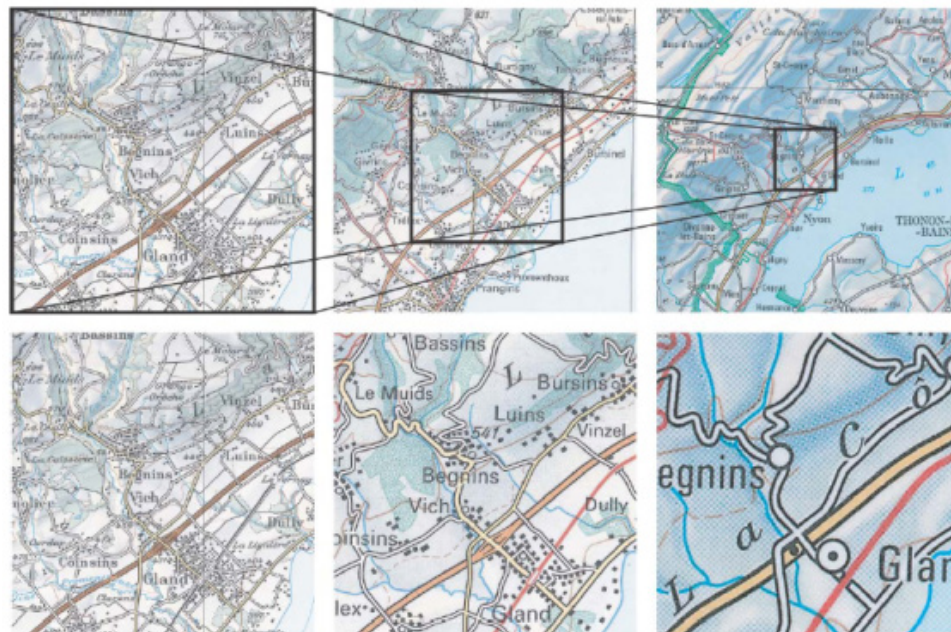


Figure 2.2 Requirement of generalization of a map: scale on the first row left 1 : 100K, middle 1 : 200k and right 1 : 500K. Maps in the second row are not to scale, from Cecconi (2003).

The complexity of manual cartographic generalization derives from its holistic and artistic nature when it comes to deciding how to portray which important aspects of information are to be preserved while omitting unwanted information (Dunkars, 2004). Attempts to automate map generalization in cartography have a long history that moves back to the advent of computers. Such attempts have not been entirely successful, and the map generalization is still an interesting topic in the scientific research community for the reason that the full automation of generalization is not yet realised (Oosterom, 2009). In addition to its holistic and artistic nature, there are many other reasons as to why generalization is so difficult to automate fully. Among these are: (a) the contextual nature of the spatial information and (b) the maintenance of topological relationships among the features on generalization and (c) its subjectivity. McMaster (1987) notes that: “Even the most skilled manual cartographers would have trouble precisely replicating their results from one day to the next”.

2.2.1 Knowledge acquisition for generalization

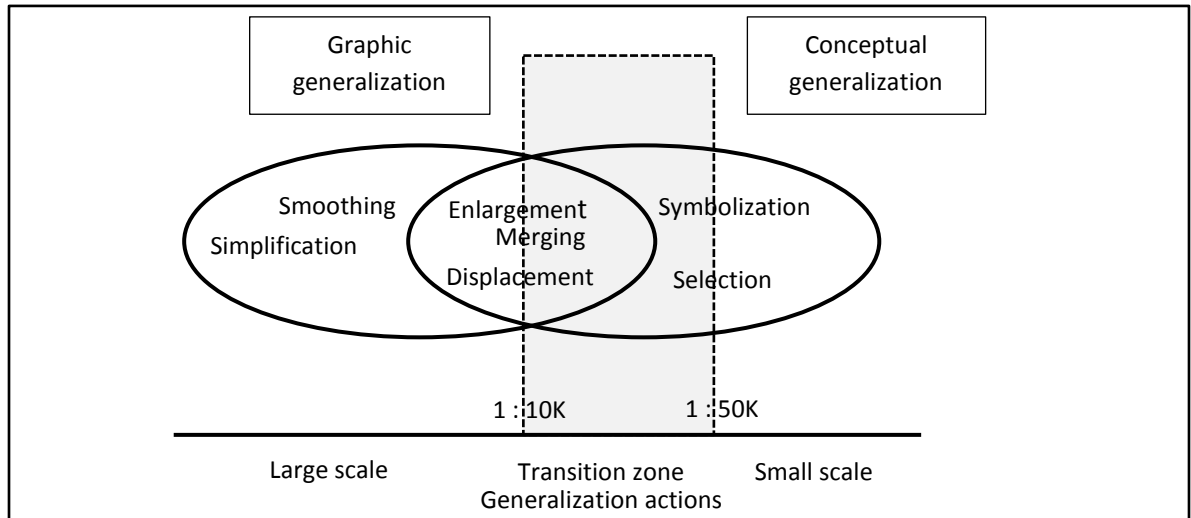
The lack of success in fully automated map generalization is due to two problems: (a) sufficiently detailed and accurate knowledge does not exist to solve generalization problems, also termed the knowledge acquisition bottleneck, and (b) difficulty in formalising the generalization process in the knowledge acquisition task mainly due to the contextual nature of the objects subject to generalization with mixed topological and semantic relations (Weibel *et al.*, 1995). Another reason given by Weibel and Dutton (1999) is “Cartographic knowledge is different from other knowledge types (e.g. the knowledge needed in medical diagnosis) in that it is essentially graphical and, therefore, hard to verbalise and formalise”. In the literature, there are several classifications given to formalising generalization knowledge (Armstrong, 1991; Kilpelinen, 1997; Ormsby and Mackaness, 1999). Among these classifications, the most influential forms of knowledge in terms of rules are topological rules, geometrical rules, semantic rules, procedural rules and contextual rules. Geometrical rules refer to the shape, size and location of objects; topological rules relate to the relationship of objects before and after generalization (e.g. a building should not cross a road after generalization); semantics refer to the meaning of

objects; procedural rules determine the generalization operators and algorithms to use and contextual rules define the rule of thumb of topographical terrain properties. For example, aggregation of buildings should not be performed if there is a canal between them. An understanding of the generalization knowledge in terms of constraints rather than rules is very important in the generalization process as explained in Section 2.2.6 below. Classification of constraints in the context of map generalization as explained by Weibel *et al.* (1995) and Weibel and Dutton (1998) consists of five categories: (a) graphical (specify size and dimensions as directed by graphical limits according to target scale) (b) topological (ensure that the relationships between features are maintained) (c) structural (define criteria describing spatial and semantic structure such as clustering and alignment) (d) Gestalt (relate to aesthetic and visual balance) and (e) process (mainly influence on how generalization operators are selected and sequenced).

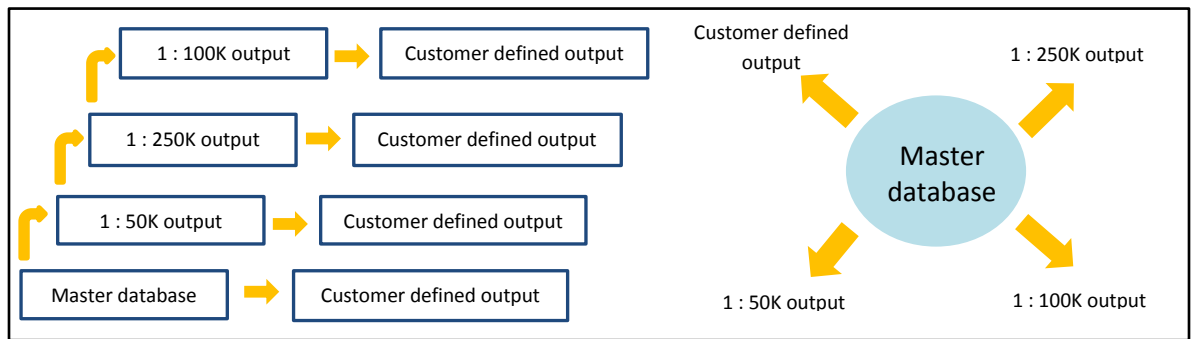
2.2.2 Conceptual architecture for generalization

Graphic and conceptual generalization

Depending on the target scale, automatic generalization can be conceptually categorised into graphic generalization and conceptual generalization in the application of generalization operations. When the scale reduction is small (e.g. 1 : 2K to 1 : 5K), a more graphic generalization that affects the geometry of objects is needed. The conceptual generalization is much more suitable when there is a significant scale reduction (e.g. 1 : 10K to 1 : 50K) where fewer details can be represented. Hence, the reduction in detail is necessary. Further, graphic generalization can be characterised by simplification, smoothing, enlargement, displacement and merging. None of these operations affect the symbology of the map whereas conceptual generalization can be characterised by merging and selection in addition to enlargement and symbolization that lead to change of classification of attributes (Krak and Ormeling, 2003). The relationship between the two types of generalization is illustrated in Figure 2.3(a) using the generalization operations. It can also be distinguished that conceptual generalization is related to the object and model generalizations while graphic generalization is related to cartographic generalization.



(a)



(b)

Figure 2.3 (a) Relation between graphic and conceptual generalization, based on Krack and Ormeling (2003) and (b) Ladder approach left and Star approach right, based on Stoter (2005).

Star and Ladder approaches

Two approaches to performing automatic generalization as illustrated in Figure 2.3(b) are: ladder approach and star approach which are often adopted by the NMAs, sometimes with a mixed approach of both according to Stoter (2005). In the ladder approach, each small scale data set is derived from a large scale data set in steps (from scale to scale). In the star approach, all small scale data sets are derived from the same base scale data set. The large to middle data sets are derived from the base data set and the smaller scale data sets are derived from one middle scale data set in the mixed approach. Deciding which approach is chosen is important depending on the application requirement. In the application of deriving several small scale data sets using model generalization, adopting the ladder approach is not suitable since it degrades spatial accuracy up the ladder. The

main purpose of model generalization is to reduce the amount of detail of a digital landscape model (DLM) at the target scale under statistical control (João, 1998). In this case, the star approach is the most suitable method. Ladder approach is more suitable for producing digital cartographic models (DCM) using cartographic generalization where the spatial accuracy is not that significant.

2.2.3 Models of generalization

With a wider meaning given in digital cartography and GIS, generalization can be considered as a process of representing the real world in different models with abstract details while preserving maximum information, depending on the purpose of the application. Generalization takes influence through creating the first model called the DLM with the use of object generalization (Figure 2.4). As a part of deriving information for specific purposes, the first DLM can be generalized in two ways, either by reducing it into a DLM of lesser information content using model generalization (statistical generalization) or by converting it into a DCM using cartographic generalization (Brassel and Weibel, 1988; Weibel and Dutton, 1999; Sarjakoski, 2007).

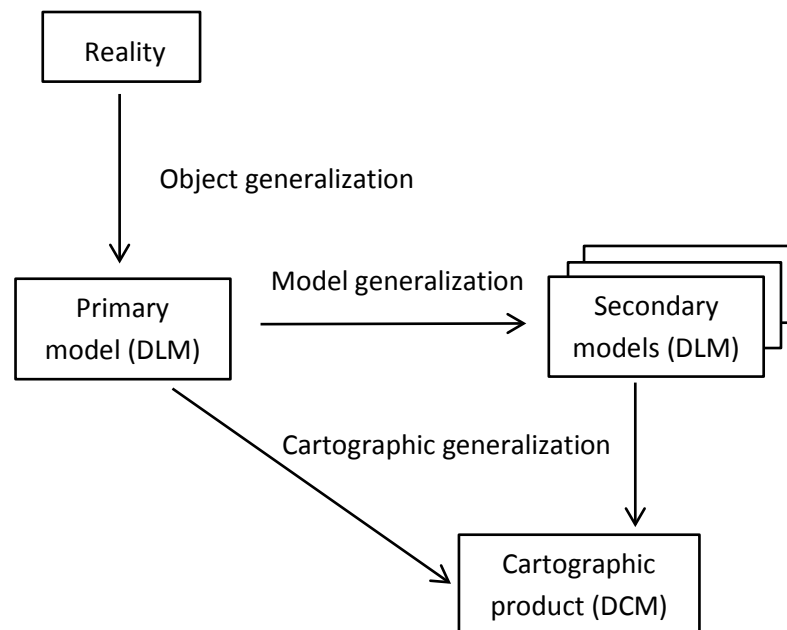


Figure 2.4 Generalization as a sequence of modelling operations, based on Weibel and Dutton (1999).

2.2.4 Generalization operators

The main task of a generalization operator is to solve a specific generalization problem. A combination of different generalization operators can be used to build up the entire generalization process or the workflow. Traditionally, cartographers use different manual operations such as selection, simplification, displacement, enlargement and combination to describe various stages of the generalization process. This manual application involves a great deal of human interpretation of spatial data and decisions about how to generalize and how to solve generalization conflicts. Therefore, a manual process is considered holistic in nature. In the digital context, it is almost impossible to develop such holistic solutions for generalization due to its subjective nature and lack of well-defined rules to guide decision-making. Therefore, a functional breakdown of the entire generalization process benefits identification of the constituents of generalization and enriches the development of specific solutions to sub-problems in the process (Weibel and Dutton, 1999). Application of the generalization operators is governed by different criteria to solve a specific generalization problem in a given situation. Even the same generalization operator implemented with different algorithms may behave differently. The use of generalization operators is influenced by three main elements: (a) the feature class (road, building, relief etc.) (b) preceding situation analysis and (c) the map scale (Cecconi, 2003). As discussed by Cecconi (2003), there are basically two types of generalization operators acting upon spatial objects:

- Independent: This kind of operator is applied to individual objects or a group of objects independent of their spatial context (not necessary to check spatial relations with feature classes, e.g. simplification, smoothing and collapsing).
- Contextual: Context-dependent operators such as selection, aggregation, enlargement and displacement can only be triggered and controlled following spatial relations with other objects. For example, a displaced building should remain on the same side of the road after generalization. If the context between different spatial objects is ignored, spatial relations between objects are lost causing topological conflicts between objects.

Several researchers have discussed and used generalization operators for geospatial mapping applications (Beard, 1987; McMaster and Shea, 1992; Robinson *et al.*, 1995; Peng, 1997; Cecconi, 2003; Galanda, 2003). However, a significant difference can be found in the literature on the number of generalization operators, and in the terminology used to describe them as observed by Rieger and Coulson (1993). Galanda (2003) has made a clear graphic and textual overview of his generalization classification of operators while Cecconi (2003) presented the classification of operators defined for the automatic generalization new technology (AGENT) project (Figure 2.5). This classification of generalization operators builds the base for the work in this thesis since it provides a detailed and comprehensive explanation of the use of such operators on both individual and/or groups of spatial objects in the application of the model generalization and the cartographic generalization.

2.2.5 Generalization algorithms

While a generalization operator defines and performs some transformation in the generalization process, a generalization algorithm is used to implement the particular transformation. A particular algorithm implementation can be based on either raster-based or vector-based theories. Many generalization algorithms implement one or several operators. For example, the building amalgamation algorithm on polygon features by Regnauld and Revell, (2007) consists of simplification and enlargement operators in addition to merging operator. Also, the line simplification algorithm by Wang and Müller (1998) is a combination of selection and exaggeration operators. There are countless generalization algorithms written over the past four decades, and more algorithms are described in the AGENT technical report by Bader *et al.* (1999).

Attribute transformation		Classification	Thematic selection		Select a subset of feature classes that are relevant to an application.
			Thematic aggregation		Changing thematic resolution (moves along a classification hierarchy).
Spatial transformation	Individual objects	Simplification	Weeding		A representation of the original line using a subset of its initial coordinates, retaining those points which are considered to be most representative of the line.
			Unrestricted simplification		A simplified representation of the original line is computed, instead of using a subset of initial coordinates, the new line may choose any point of the space and may even consist of many points.
		Collapse			The decomposition of features of n dimensions in features of n-1 or even n-2 dimensions.
		Enhancement	Enlargement with regard to geometric constraints	Enlargement	Constant enlargement of all directions (scaling).
				Exaggeration	Exaggerate important parts – Enlargement with change of shape.
			Enhancement with regard to semantic constraints	Smoothing	Change of geometry of an object to improve the aesthetic quality
				Fractalisation	
				Rectification/ Squaring	Rectify the geometry of objects which are expected to have a rectangular shape.
		Individual objects or set of objects	Selection/ Elimination	Selection	Select the most important object from a cluster/ network to represent the original feature.
	Elimination			Eliminate unimportant objects from the map.	
	Displacement		Move objects to solve conflicts between objects that are too close or to keep important neighbourhood relations eg. if a bend is moved through filtering, a road next to the building also has to be moved.		
	Set of objects	Join features to one object	Amalgamation	Fusion	Aggregation of the two connected objects of the same nature
				Merge	Join disconnected objects
		Aggregation	Combine	Combine a set of objects into one object of higher dimensionality	
			Typification	An initial set of objects is transformed into a new generalized group. It is not clear after the transformation which original object(s) created a new one - the new objects are merely space holders	
		Join features to several objects		The initial group might be built of disjoint objects (such as buildings) or be created through segmentation of one single object (such as road segments). The former type is called structuration, the latter one schematisation.	

Figure 2.5 Generalization operators defined for the AGENT project simplified after Bader *et al.* (1999), based on Cecconi (2003). Note: Different colours are only used to enhance the visual clarity.

2.2.6 Review of the generalization frameworks

Automatic cartographic generalization has been used to find solutions to restricted spatial problems, in particular for linear features by Douglas and Peucker (1973) and point and area features by Töpfer and Pillewiser (1966). According to Buttenfield and McMaster (1991), the first conceptual framework for automatic map generalization has been presented by Ratajski (1967) who identifies two fundamental types of generalization processes: (a) quantitative generalization which consists of a gradual reduction in map content depending on scale change and (b) qualitative generalization which results from the transformation of elementary forms of symbolization to more abstract forms. After Ratajski's model, several frameworks for automatic map generalization have been proposed, which broadly follow two models: (a) process oriented model which structures the entire generalization process by the process of structure recognition and (b) object-oriented model which addresses the level of map objects as distinguished by Steiniger and Weibel (2005). One of the most influential process-oriented models is the framework proposed by Brassel and Weibel (1988) where five stages are distinguished in the process: (a) structure recognition (b) process recognition (c) process modelling (d) process execution and (e) data display. Process modelling involves the compilation of rules and procedures in generalization. Among the drawbacks in adopting rules is the difficulty of acquiring and formalising cartographic knowledge; a requirement for a great amount of rules to describe conditions and actions between map objects, and sequencing of generalization operators since they affect each other.

A major step forward in dealing with the generalization frameworks is the introduction of object-oriented models and their associated spatial functions into automatic map generalization where a constraint-based modelling approach is adopted as proposed by Beard (1991). Constraints formulate a synthesis of conditions that a generalized map should adhere to find a goal state in which a variety of constraints will be satisfied. However, in contrast to rule-based modelling, violation of a single condition is not bound to an individual action to solve a particular generalization problem in this type of modelling. The constraint-based modelling approach by Ruas and Plazanet (1996) using an

object-oriented framework is a combination of the model of Brassel and Weibel (1988) which comprises of parts of the structure recognition and process recognition as well as the process modelling and execution, and the constraint-based and iterative refinement strategy suggested by Mackaness (1995) with trial and error (backtracking facilities) to generate a satisfactory generalized map. The proposed framework has been implemented in an experimental stage at IGN France with three levels of processing where the highest level being the 'global master plan' that determines a sequence of generalization tasks to apply to the entire map (e.g. aggregate all connected objects of the same class). The second level is the choice of the geographical region according to the given task (e.g. the objects are contained in an urban block). The third and final level called the local level is the development and execution of a generalization plan for every situation with the generalization output being evaluated at the end. These generalization plans locally determine generalization operations (e.g. simplification, smoothing) and their sequence while considering appropriate algorithms and parameters for such operations (e.g. line simplification with the Douglas Peucker algorithm (Douglas and Peucker, 1973) with a constraint threshold of 5m distance). At this local level, if the generalization evaluation phase reports an unacceptable output, three types of actions are possible: (a) selection of other parameter values (b) use of another algorithm or (c) a new local plan is chosen. This refinement process is continued until a satisfactory result is obtained at the evaluation phase (Steiniger and Weibel, 2005). This constraint-based model has been further refined and re-implemented in the prototype generalization system of the AGENT project (Barrault *et al.*, 2001) with its commercial successor software - CLARITY - by Laser-Scan (Figure 2.6).

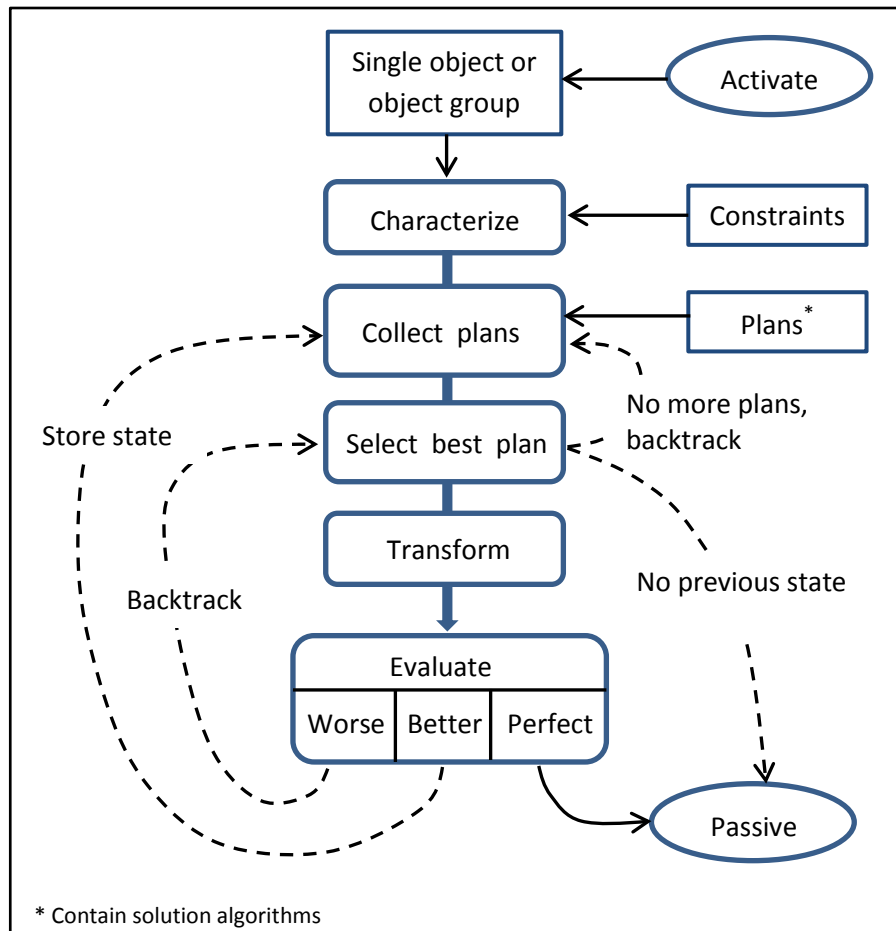


Figure 2.6 AGENT generalization procedure for a single object or a group of objects, based on Steiniger and Weibel (2005).

The generalization model used in the German ATKIS (the Official Authoritative Topographical Cartographic Information System) project as mentioned by Brassel and Weibel (1988) and Morgenstern and Schürer (1999) represents a different view of generalization, which identifies three distinct generalization processes as described in Section 2.2.3 above. The first type is object generalization which describes a mental generalization process in the sense of abstraction and selection from the real world data by the data collecting person (land surveyor, aerial photo analyst or GIS data analyst). The second type of generalization - model generalization - can be seen as a pre-processing step prior to visualisation through cartographic generalization (Sarjakoski, 2007). This process is also called statistical generalization (Brassel and Weibel, 1988) because it is mainly a filtering process after object generalization, aimed at data reduction under a certain statistical control to maintain quality of information as much as possible with generalization

operators such as selection, removal and reclassification operations with more or less no conflicts between objects. It also causes lesser generalization effects in which reduction of data volume is maximised while, at the same time, modification of data is minimised (Weibel, 1992). Further, it aims at the minimum average displacement of objects. As a result, the statistical generalization mainly affects the symbology of the map to create a minimum of conflicts. Cartographic generalization being the third type, aims to modify the local structure of the data, and hence is non-statistical (Brassel and Weibel, 1988). In this process, the main aim is to give a better visual effectiveness to the graphic display with the best use of map space to optimise legibility at a given scale for a particular purpose of the map. This would more often cause graphical conflicts to be resolved after generalization using the operations such as symbolization, enlargement, displacement and exaggeration.

2.2.7 Relationship of generalization models to wayfinding maps

The selection of the desired objects and the features as well as the desired resolution of the presentation is a matter that needs to be highly focused on processing and viewing data in wayfinding maps on mobile devices due to their limited display size, resolution and processing power (Hampe, Anders and Sester, 2003). As a result, a new technique called adaptive visualisation has come to the fore in handling wayfinding maps as discussed in Section 2.1 above. For this purpose, the advantage of a multiple representation database (MRDB) has been discussed by Hampe, Anders and Sester (2003), Hampe and Sester (2004), Hampe, Sester and Harrie (2004) and Elias, Hampe and Sester (2005) in the adaptive visualisation of geographical information on wayfinding maps. An MRDB is a spatial database used to store the same real world phenomena at different levels of precision, accuracy and resolution (Devogele *et al.*, 1996; Weibel and Dutton, 1999). According to Hampe and Sester (2004) an MRDB is mainly characterised by two features: (a) different levels of details are stored in one database and (b) the corresponding objects at different levels are linked (Figure 2.7). When establishing an MRDB, there are two possibilities according to Hampe, Anders and Sester (2003): (a) linking existing data sets at different resolutions by data matching procedures if data across these resolutions are

consistent and (b) creating new data sets from the existing data sets with the links in the application of the automatic map generalization.

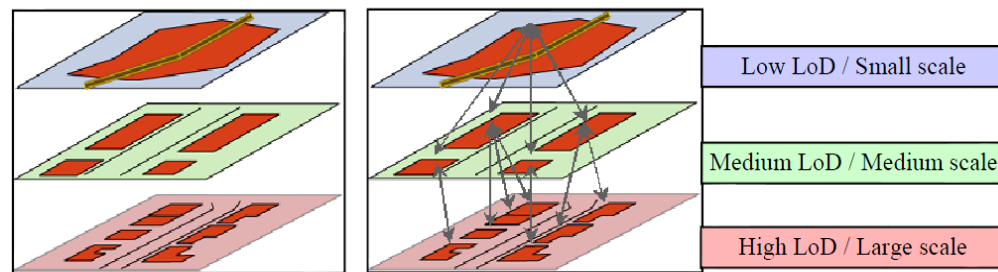


Figure 2.7 Characteristics of an MRDB - storage of multiple representations of objects (left) and linkage of corresponding objects (right) from Hampe and Sester (2004).

One of the main advantages to NMAs in establishing MRDBs using existing vector data sets at different resolutions, represented either in a DLM or a DCM, which have been evolved from the development of generalization in the NMAs according to Stoter, Kraak and Knippers (2004), is the storage of pre-generalized data involving complex time consuming algorithms at different levels for subsequent use. As a result an MRDB has enabled the adaptive visualisation of wayfinding maps used in mobile devices because real-time processing of these complex generalization algorithms in order to generate views at any desired scale of the mobile user has still not fully been realised (Hampe, Anders and Sester, 2003). However, real-time generalization can be efficiently performed in regions with limited scales using the generalization algorithms having less complexity. The notion in such instances is to apply the real-time generalization to data stored in an MRDB at a scale which is close to the desired scale chosen by the user (Hampe, Anders and Sester, 2003; Hampe, Sester and Harrie, 2004). When the scale change is small, reduction of data content is small and, therefore, more involvement of graphical generalization is required as mentioned in Section 2.2.2 above.

However, in a wayfinding scenario with focus maps, what is most required is an eye-catching distinction between the prominent landmarks and the background features. As a result, generating views with the real-time generalization within a small scale change is less significant. In such instances, the use of multiple representations of relevant features at two desired representation levels with a significant scale difference is an ideal situation

(representation between large scale and small scale data sets in an MRDB). One such example is the representation of buildings at two scale levels in an MRDB by Elias, Hampe and Sester (2005). In this example, buildings at smaller scale have been derived by aggregating building groups into a single entity using an aggregation generalization operator, which is more an application of the statistical generalization as discussed in Section 2.2.6 above, aiming at reduction of data at the smaller scale based on conceptual generalization. Further, minimising of topological conflicts in generalized data while maintaining its accuracy in the statistical generalization is an added advantage to preserve the quality of information on the generalized features with the least displacement.

2.2.8 Related work on building geometries

In the focus map representation for wayfinding, one of the key aspects is the application of the object-directed transformation technique which is the method of reducing details on the map using map generalization techniques. Different possibilities in highlighting important objects such as landmarks have been discussed in the literature using either graphical variables such as colour and opacity, representation of objects with variable scale techniques and/or generalization with simplifying features. As discussed, there have been attempts to emphasise the focus by overlaying the important building on the merged background building group, that is, merge background buildings while leaving the salient landmark building separate (Sester, 2002) and emphasising the salient landmark building by replacing the corresponding merged background group of buildings (Elias, Hampe and Sester, 2005). In these instances, the emphasis of the salient landmarks is made distinct from the coarse background information. Therefore, aggregation of building groups to be treated as merged background information in map generalization plays an important role in emphasising landmarks for wayfinding.

Aggregation of building vector data using displacement and then rotation of one building in the adjacent pair of buildings to align the two buildings before merging has been discussed by Lichtner (1979), Ware, Jones and Bundy (1995) and Jones, Bundy and Ware (1995) using triangulation (Figure 2.8). However, their approaches cannot deal with the pairs of buildings, each positioned in exceptional locations such as almost overhanging

and total separation locations as illustrated in Figure 2.9 since merging of buildings in such pairs produce corner touching building entities.

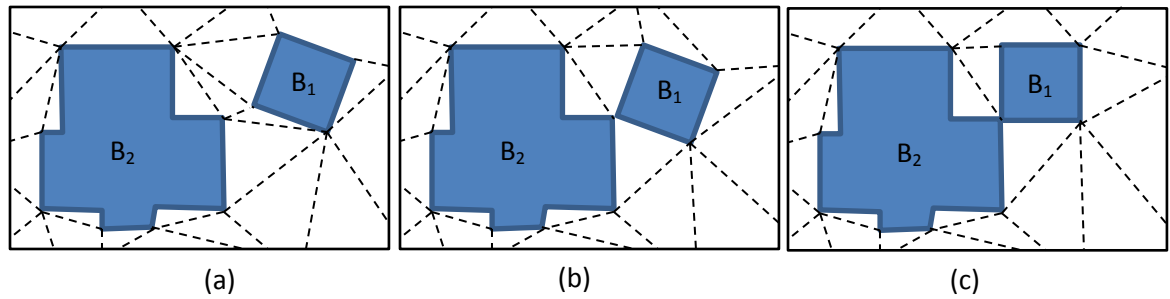


Figure 2.8 (a) A pair of buildings at total separation (b) displacement of building B₁ to building B₂ has created a corner touching situation and (c) rotating, aligning and merging of building B₁ with building B₂, has created a corner touching situation.

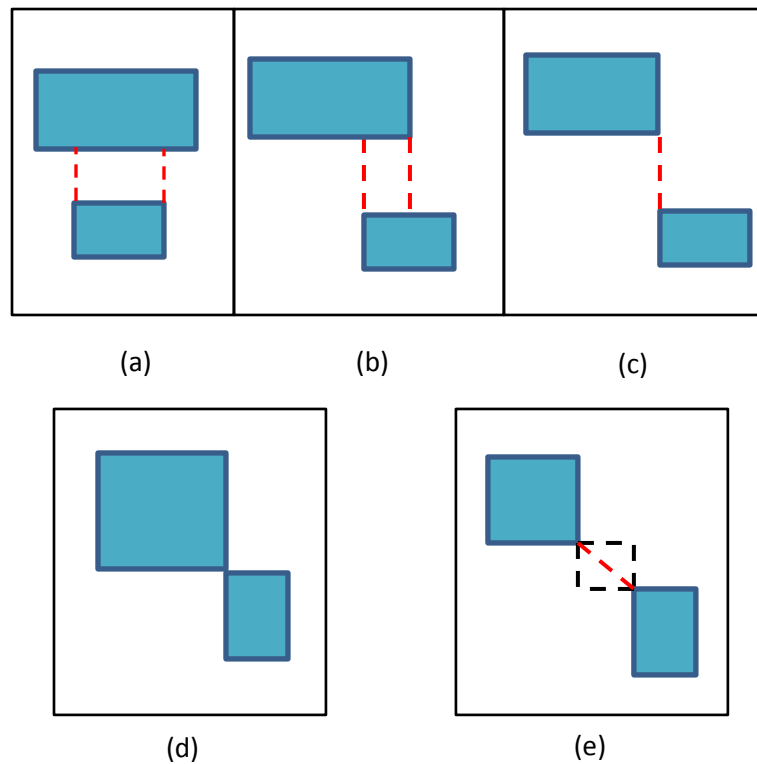


Figure 2.9 A pair of buildings in a cluster at different positions: (a) total overhanging (b) partial overhanging (c) almost overhanging (d) corner touching and (e) total separation, based on Regnauld and Revell (2007).

The approach used by Schylberg (1992) on raster data for the aggregation of buildings comprising of orthogonal sides based on dilation and erosion also cannot deal with the

building objects at exceptional locations as explained above to bring out a meaning to the merged building. Further, the approaches by Li (1994) and Su *et al.* (1997) based on mathematical morphology on raster data for building aggregation have the same problem in dealing with buildings placed in three exceptional locations described above. In the approach of Su *et al.* (1997), depending on the target scale, the size of the structuring element that is used in filling the gaps between buildings in the cluster is calculated based on an equation. However, this does not guarantee that all the gaps are filled in at a specific scale as evident from Figure 2.10(b) when compared with source cluster in Figure 2.10(a) where the building at the bottom of the cluster is not aggregated. Ai *et al.* (2007) have used a method of filling gaps between building clusters using rasterizing techniques and then converting the rasterised amalgam into the vector. However, they have not explicitly mentioned how they have dealt with clusters where buildings are placed in exceptional locations as discussed above.

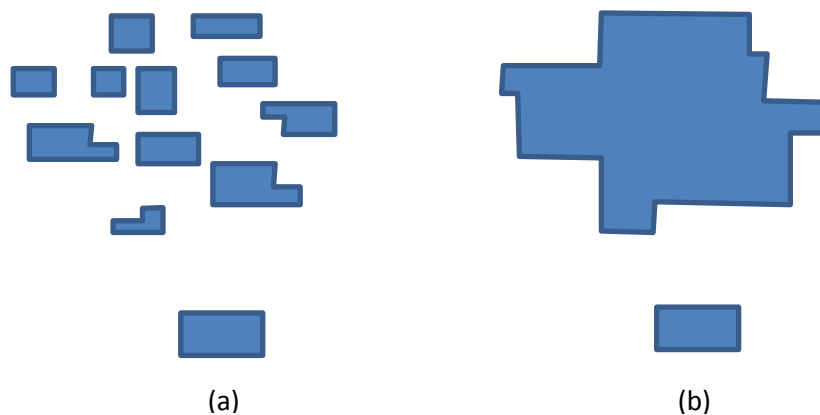


Figure 2.10 Area aggregation based on morphological operators: (a) source building cluster and (b) amalgam after aggregation with a structuring element formed at 7 x scale reduction where the bottom building is not aggregated; from Su *et al.* (1997).

Regnauld and Revell (2007) have described an approach to aggregate building vector geometries to give a single orthogonal building by squaring each building in the group and then filling the gap between each pair of buildings. This gap is chosen in the order of the minimum spanning tree (MST) that runs with the weight of the minimum distance between each pair of buildings in the group for the five types of exceptional building locations they have identified - 'total overhang', 'partial overhang', 'almost overhanging', 'corner touching' and 'total separation' - as illustrated in Figure 2.9. This filling method

almost follows the configuration of the series of structural elements used by Su *et al.* (1997) for shape refinement of the merged building. Although their method has also come out with a shape refinement of the merged building through the simplification of granular sides and the enlargement of juts, a separate algorithm in the 1Spatial generalization platform - CLARITY (Revell, 2008) - is used to enlarge narrow corridors created as a result of aggregation.

Savino (2011) has developed a method to aggregate buildings, creating an initial outward buffer with a minimum distance chosen based on the target scale around each building in the data set (e.g. building B0 in Figure 2.11), and retrieving buildings that intersect with the initial outward buffer. In this method the gaps between the building with initial outward buffer and the retrieved buildings (buildings 1, 2 and 3 in Figure 2.11) are merged with the oriented bounding rectangles of the convex hulls formed using the intersection between each outward buffer of the retrieved buildings and the initial outward buffer. The oriented bounding rectangles are generated based on the method by Regnauld and Revell (2007). However, if the area of the convex hull is less than a particular area threshold between two buildings, the two buildings are considered too far away and not glued with the oriented bounding rectangle (e.g. building 3 is not glued as shown in Figure 2.11(e)).

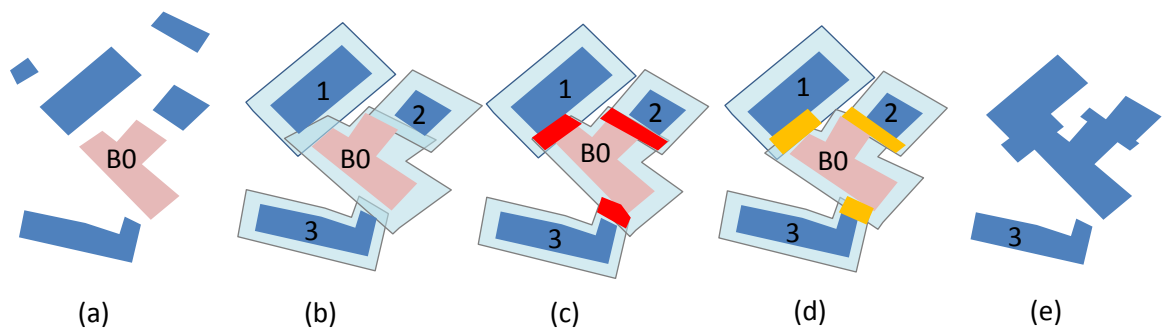


Figure 2.11 Building aggregation: (a) initial building B0 subjected to buffering based on threshold distance (b) Buffering of buildings 1, 2 and 3 that intersect the initial building B0 with the same threshold distance (c) convex hulls (d) oriented bounding rectangles between building B0 and 1, 2 and 3 and (e) amalgam created by gluing buildings with the oriented bounding rectangles where oriented bounding rectangle between building B0 and 3 is ignored considering the area of the convex hull between buildings B0 and 3, based on Savino (2011).

Several algorithms that deal with a generalising outline of either single buildings or amalgams exist in the literature. The algorithm by Julian (2009) squares the edges of a building polygon, taking into account an orientation threshold based on the wall statistical weighting algorithm by Duchêne *et al.* (2003). Further, Rainsford and Mackaness (2002) have developed a squaring algorithm using template matching on predefined shapes. Lamy *et al.* (1999) have discussed an algorithm for squaring and enlarging narrow sections (corridors) with the use of the AGENT technology. Revell (2008) has mentioned an algorithm implemented in the CLARITY software for enlargement of narrow corridors although the algorithms used in CLARITY are not given in the literature. Sester (2000a) has developed an algorithm for simplification of orthogonal buildings. Further, Fan and Meng (2010) have developed an algorithm to simplify buildings which have non-orthogonal sides.

When aggregating natural objects (e.g. buildings with irregular shape) as opposed to rectangular building objects, filling spaces between each pair of objects with triangles formed using the constrained Delaunay triangulation (CDT) has been presented by Jones, Bundy and Ware (1995) and Ware, Jones and Bundy (1995). These filling triangles are selected so that the edge distance between triangles in a pair of objects does not exceed some distance threshold based on the related technique described by DeLucia and Black (1987).

2.3 Data enrichment

Data enrichment is the process of enhancing, refining or improving raw data to make data a valuable asset pertaining to a specific use or application. One of the main applications of data enrichment in this research is to derive building clusters (groups) to be used in the building aggregation process for the subsequent map generalization as discussed in the previous section. Another important application is to derive salient landmarks under data mining which is the discovery of knowledge from existing data for decision-making about complicated processes. A simple example of data enrichment is harvesting fish from the ocean (Pyle, 1999). In order to perform this task, there are three tools existing with the advancement of technology: (a) data modelling (that reveals each 'fish') (b) data mining or

surveying (that looks at the area of the ocean for fish and is the ‘fish finder’) and (c) data enrichment (that removes murk and clears the water so that ‘fish’ can be clearly seen and easily attracted). Data enrichment can be used in both attribute and spatial data. For example, bank account details of the general public in a data store are a type of attribute data set while a set of building geometries in a spatial database contains both attribute and spatial data. For example, automatic map generalization involves the use of several integrated generalization algorithms in a comprehensive workflow which is a decision-making process and requires some additional information about data to apply the algorithms correctly in the workflow. Further, in spatial data mining the execution of decision-making algorithms requires exploring the information that are hidden in the raw data. Generally the spatial databases with the raw data could not support decision-making. One reason is the lack of auxiliary information derived from the spatial data structures to describe spatial relationships within the data. Therefore, data enrichment is necessary to equip raw data with additional information (auxiliary data) with their semantics, geometry and spatial relationships and common patterns for subsequent automatic map generalization and data mining processes.

According to the literature, spatial data structures that are of much importance to extract auxiliary information include Delaunay triangulation (DeLucia and Black, 1987; Jones, Bundy and Ware, 1995; Ware, Jones and Bundy, 1995; Jones and Mark, 1998; Li *et al.*, 2004; Haowen, Weibel and Bisheng, 2008; Qi and Li, 2008), Voronoi diagrams (Basaraner and Selcuk, 2004; Haowen, Weibel and Bisheng, 2008), MSTs (Regnauld, 2001, 2005; Qi and Li, 2008), graph structures (Mackaness and Beard, 1993; Regnauld, 2003, 2005), Skeletons (Bader and Weibel, 1997), hierarchical partitioning schemes (Ruas, 1995) and machine learning (Sester, 2000b).

2.3.1 Relations in data enrichment

Data enrichment has two main relations: (a) horizontal and (b) vertical as identified by Neun, Weibel and Burghardt (2004). Horizontal relations exist on the same level of detail in a data set and represent common structural properties such as proximity, topology, pattern and alignment, and helps recognise and derive clusters of geometrical objects,

while vertical relations can exist among homologous objects or groups of objects such as shape, size, compactness, object links and scale relations. Identification of horizontal relations in data sets not only helps to identify structural knowledge required to deal with the contextual nature of map generalization, but also to leverage horizontal relationship among data sets in a vertical relation at different resolutions to represent the same phenomena. These vertical relations can exist on both attributes and geometric features. Vertical relations are important to leverage structural knowledge (horizontal relations) of different levels of details. In vertical relations, the object links (object identification numbers) of different data sets can be integrated and maintained in an MRDBMS (Hampe, Anders and Sester, 2003; Sarjakoski, 2007). Links needed for the integration of such data sets can be maintained either in one separate table or among the tables of the data sets according to Hampe, Anders and Sester (2003).

2.3.2 Clustering

Clustering is the process of grouping similar objects based on their geometric, spatial and semantic characteristics and is a technique for data interpretation. According to Anders (2003), there are basically three types of clusters: non-hierarchical clusters, hierarchical clusters and graph-based clusters. Non-hierarchical clusters create a simple partitioning of a data set into a set of k non-overlapping clusters where each cluster must contain at least one data element, and each data element must belong to exactly one group. The hierarchical clustering represents a tree view with multiple levels of partitioning where on top is a single cluster which includes all other clusters. At the bottom are the clusters with single elements. The tree can be constructed either top-down (divisive approach) or bottom-up (agglomerative approach). In the agglomerative approach the two most similar objects are merged together based on similarity measures in the subsequent steps and as a result of merging, the total number of clusters is decreased by one. These steps are repeated until only one large cluster is left, or a given number of clusters are obtained or the distance between the two closest clusters is above a threshold. The divisive approach works in the reverse direction starting with a large cluster. The graph-based clustering initially involves computing neighbourhood graph (Jaromczyk and Toussaint, 1992) such as an MST

of the original points. Then any edge in the graph that is much longer than its neighbours is deleted according to certain criteria to form forests which are trees representing clusters. The result of forests can be represented in a similarity matrix such that every element of the matrix represents the similarity between two objects (Anders, 2003).

2.3.3 Related work on clustering building geometries

Data enrichment has been used over the last two decades to enhance geometric, spatial and semantic properties of features with auxiliary information for subsequent automatic map generalization to assist with the use of computationally intensive generalization algorithms in order to reduce complexity. The following paragraphs describe the research undertaken to cluster buildings into groups, considering properties of geometric features.

Regnauld (1996, 2001) has conducted research to cluster buildings using an MST which is an algorithm to create a neighbourhood graph with nodes and edges (Anders, 2003) using the proximity between the buildings as the weight. In order to create building clusters, the MST is segmented based on calculating the factor of inconsistency proposed by Zahn (1971) with the help of average weights and standard deviation of nearby edge weights of both ends of a particular edge in the MST. Further analysis of the homogeneity of groups has been carried out applying only the criteria of size and orientation of buildings in the Gestalt theory. In this case the shape of buildings which is an important criterion to identify groups has not been considered since the process has had more focus on generalising areas of dense individual housing where the shape is rarely considered as a defining characteristic.

Steinhauer *et al.* (2001) have designed and implemented an approach to cluster buildings into groups to form abstract regions on cartographic maps using the following criteria: adjacency of buildings, proximity between buildings and the cardinality of buildings with the generation of Voronoi diagram and convex hull of all the buildings. Christophe and Ruas (2002) have developed an approach to detect buildings aligned in rows using a series of simple hypothetical straight lines from a single anchor point located at the x-min and y-min of the spatial features of a block. The advantage of this approach is that it does not

require the development of complex spatial structures such as the MST, the Delaunay triangulation and the Voronoi regions. This approach comprises of two steps: the first step is to identify candidates of aligned buildings within the block and the second step is to filter the perceptual alignments out of the candidate alignments using the constraints of buildings such as proximity of building alignment, arrangement of buildings (quality of sides and centre of buildings) in alignment and similarity of buildings in the form of size, shape and wall orientation.

Data enrichment approaches by Regnauld (2003, 2005), Li *et al.* (2004), Revell, Regnauld and Thomas (2005), Ai *et al.* (2007), Regnauld and Revell (2007) and Haowen, Weibel and Bisheng (2008) have adopted clustering of building objects based on the Gestalt factors using the Delaunay triangulation for subsequent automatic map generalization. However, variations in the methods used to derive the Gestalt factors for such clustering are found in Li *et al.* (2004), Ai *et al.* (2007) and Haowen, Weibel and Bisheng (2008). Clustering approaches discussed by Regnauld (2005) and Regnauld and Revell (2007) are based on mapping adjacency relations in terms of the minimum distance between buildings, between roads and between buildings and roads stored in a proximity graph. Then through edge filtering of the proximity graph on a minimum distance threshold between objects, clusters are derived (Figure 2.12).

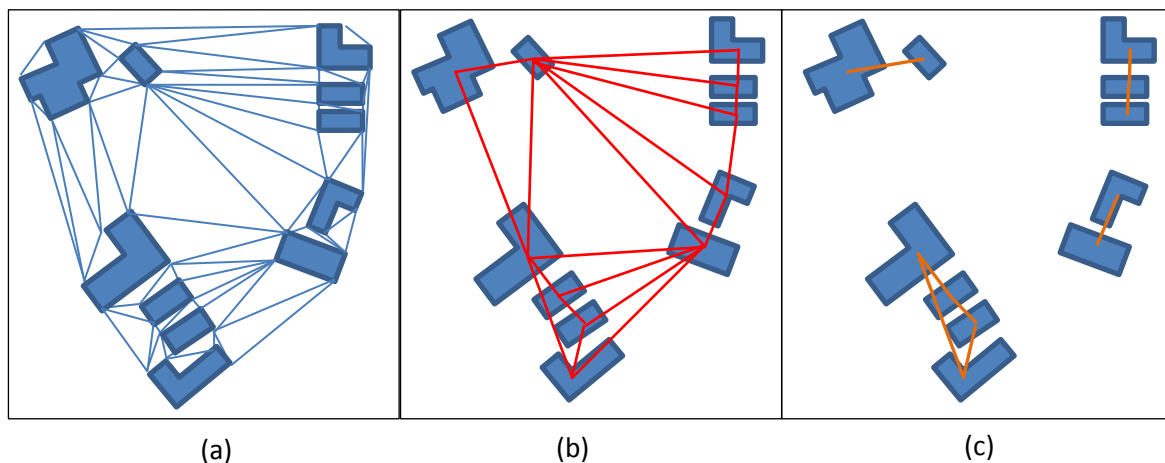


Figure 2.12 Building clustering with three steps: (a) CDT (b) connectivity links between buildings in the proximity graph and (c) four building clusters through filtering of the connectivity links between buildings based on the minimum distance threshold.

Allouche and Moulin (2005) have developed an approach to identify geometric feature clusters on maps using the Kohonen-type neural network analysis and the Delaunay triangulation for map generalization. This approach can replace cartographic elements such as buildings in a region by its surrounding polygon. The use of this type of neural network has allowed them to generate different levels of groupings with respect to various zoom scales of the map thus enabling the generation of multiple representations of the same phenomena in the context of map generalization.

Anders *et al.* (2006) have developed an approach to typify building geometries with the help of building groupings based on the identification of buildings in a grid structure format. In this approach first the grid structures are detected with the assistance of a so-called relative neighbourhood graph (RNG) which detects grid-like graph structures. Then the detected grid structures are regularised by the least-square adjustment with an affine transformation or a Helmert transformation.

Chaudhry and Mackaness (2006) and Basaraner and Selcuk (2004, 2008) have used buffering technique to derive building clusters. However, in the approach of Basaraner and Selcuk (2004, 2008), they have further used the Delaunay triangulation and the Voronoi diagrams in order to enhance the spatial relationship among clusters required for the automatic map generalization.

Although several research studies have so far been conducted pertaining to building polygon grouping to simulate the manual process with well-defined and sound theories and techniques such as Delaunay triangulation, Voronoi diagrams, graph theory, Kohonen's self-organising maps, urban morphology and Gestalt theory, such studies have not considered the hierarchical relationship between constraints other than the work by Qi and Li (2008) which refers to vertical relation in data enrichment by Neun, Weibel and Burghardt (2004). The approach by Qi and Li (2008) has employed two categories of constraints namely the contextual features and the Gestalt factors, both of which can be ordered hierarchically. Based on these two categories of constraints, building groupings can be divided into two groups: groupings based on the global contextual constraints such as road and hydrographic network; and groupings based on local constraints of the Gestalt

factors: proximity, orientation and similarity. Next the generation of the CDT (see Section 2.4.2 below) to create adjacency relationships of buildings has been carried out. The approach of creating a connectivity graph is similar to that of Regnauld (2005). Then the degree of proximity, the difference of orientation and the degree of dissimilarity of each pair of buildings are calculated and attached to the connectivity graph. Finally, the MST is used to explore the adjacency relationships of buildings of the three Gestalt factors applied sequentially to cluster buildings based on the MST segmentation (Figure 2.13). The clustering approach is very interesting in that it creates clusters top-down, identifying them initially based on proximity. It also closely follows the manual cartographic process of building grouping, considering the hierarchical relationship of buildings. However, the results have not been tested and validated using real data sets.

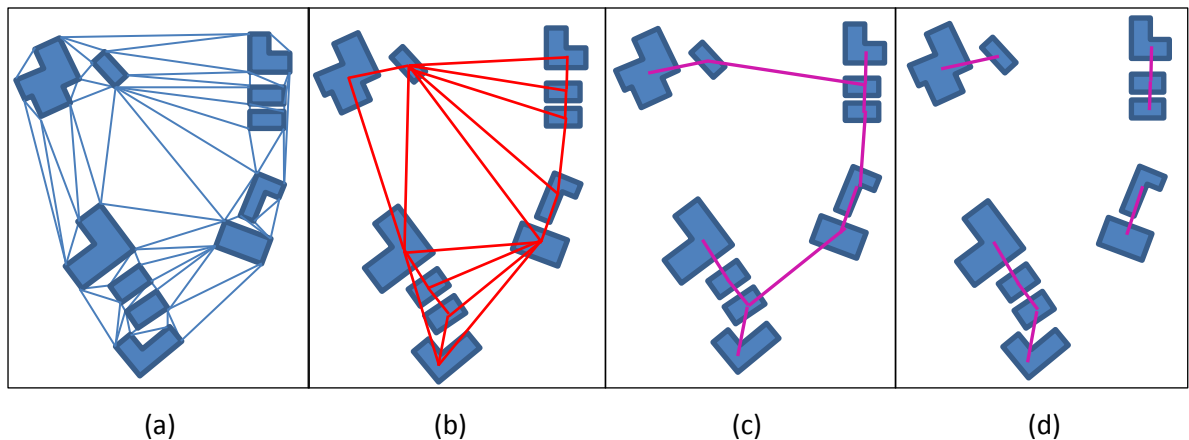


Figure 2.13 Building clustering with the MST: (a) CDT (b) connectivity graph (c) MST created from the connectivity graph with the weight of the tree as the minimum distance and (d) initial building clusters using the MST segmentation based on the threshold value of the minimum distance.

Although Li *et al.* (2004), Basaraner and Selcuk (2004, 2008), Haowen, Weibel and Bisheng (2008) and Qi and Li (2008) have considered contextual features to identify building regions or blocks for clustering with the Delaunay triangulation, Regnauld and Revell (2007) have only considered contextual features, including features with dead ends (e.g. roads) to avoid generating clusters across such features within a region or a block in the application of the triangulation. Further, none of the clustering approaches discussed above have considered semantic/thematic aspects of buildings since the resultant groups

in the clustering process are mainly based on the geometric analysis. In defining orientation, Li *et al.* (2004) have used the longest edge (major axis of buildings) as a measure while Haowen, Weibel and Bisheng (2008) and Qi and Li (2008) have used the Minimum Bounding Rectangle (MBR). The longest edge is not a good measure if the shape of the building has several orientations of sides and/or irregular shapes. Further, the MBR is not a good measure when the shape of the building is square or terraced (Duchêne *et al.*, 2003) while the MBR is consistent with buildings comprising of irregular shapes. However, the main disadvantage of all these clustering approaches is that visual perception tests for evaluating building clustering have not been carried out due to the complication of such tests.

Constraints for polygon clustering

According to Li *et al.* (2004) who have performed building clustering as mentioned in the previous section, two steps are involved in the cognitive interpretation of visual signals: a pre-attentive phase and an attentive phase. In the former, an attempt is made quickly to extract information from an image through a global search. The latter is a local phase in which attention is drawn to specific features of the visual landscape that have been identified as being different during the pre-attentive phase. It is understood that similar steps are to be adopted to visual information processing in map recognition. Therefore, it can be imagined that the cartographers must have followed the same two-step process in manual generalization of building features. Further, there is a group of global constraints and a group of local constraints for initial grouping and subsequent generalization of buildings. According to Weibel and Dutton (1998), a constraint is referred to as a design specification that a generalization problem should adhere to. These two types of constraints (local and global) have been already adopted in the approach of automatic generalization of buildings (Li *et al.*, 2004; Haowen, Weibel and Bisheng, 2008; Qi and Li, 2008). The global constraints can be considered as contextual features which mostly comprise of roads and rivers often used to partition buildings into groups due to their network structures. The local constraints, namely the Gestalt factors are from the Gestalt theory which is the study of criteria influencing grouping perception. Wertheimer (1923)

has laid the foundation for identifying a number of important perceptual principles (criteria) for object grouping. These principles are: (a) proximity (b) similarity (c) closure (d) continuity and (e) common fate. These principles are called the Gestalt laws of perception according to Thórisson (1994). To the criteria list of Wertheimer (1923), some more criteria have recently been added: common region (Palmer, 1992), element connectedness (Palmer, S. and Rock, 1994), density change and concentration (Sadahiro, 1997), common orientation (Li *et al.*, 2004) and directional relations (Haowen, Weibel and Bisheng, 2008). Among these Gestalt factors, proximity, common orientation and similarity are relevant to the spatial distribution of buildings.

Hierarchy of constraints

As discussed above, the two categories of constraints - global and local - guide the process of building grouping, and the final grouping results vary depending on the outcome of both categories of constraints. When observing the manual building grouping process for generalization by the cartographers, the use of constraints conforms to the human's customs of spatial cognition (Qi and Li, 2008). Information on maps is arranged hierarchically, and hierarchical classification methods are used for reasoning and clustering (Hirtle, 1985; Timpf, 1999). Cartographers generally adapt hierarchical constraints in the manual building grouping process for subsequent generalization. First, roads and river networks are used to partition the area represented on the map into different regions and then for each region; the Gestalt factors are employed to further partition buildings into groups. Figure 2.14 represents the constraints of building clustering with their hierarchical relationship according to Qi and Li (2008).

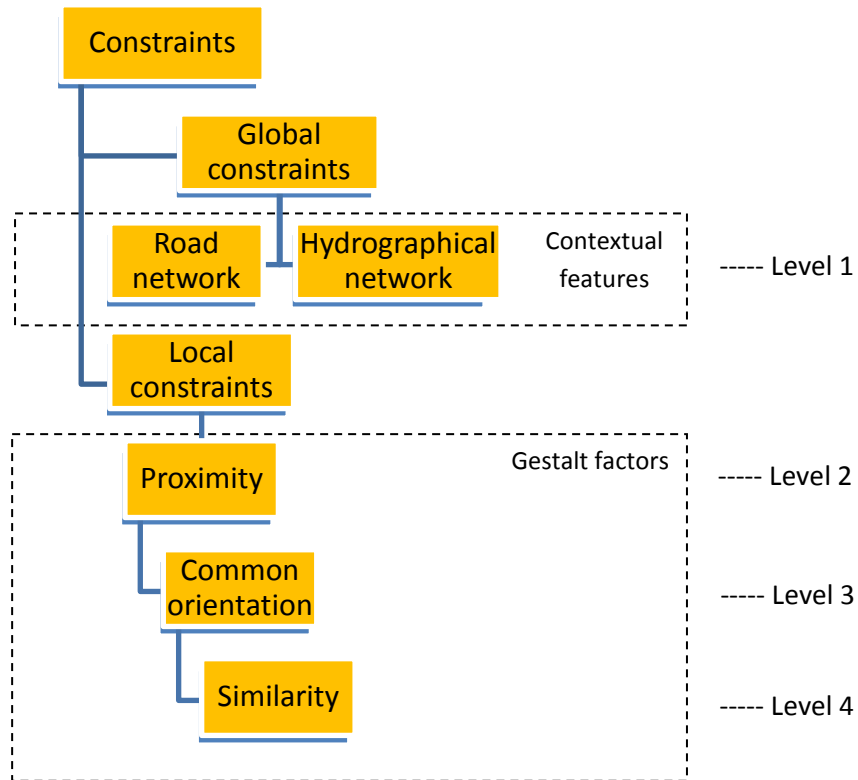


Figure 2.14 Hierarchical relationship of constraints for building clustering, based on Qi and Li (2008).

2.3.4 Related work on data mining

Data enrichment techniques are involved in the extraction of the important attributes (auxiliary information in terms of geometric, spatial and semantic data) from spatial databases that are required for the subsequent salient landmark generation using data mining processes (Raubal and Winter, 2002; Brenner and Elias, 2003; Elias, 2003; Nothegger, Winter and Raubal, 2004; Elias, Hampe and Sester, 2005). However, no work has been found in the literature as to the use of data enrichment in the context of extracting attributes (especially geometric and spatial attributes) from spatial databases using spatial data structures, such as triangulation, for the derivation of salient landmarks with the automatic methods and tools.

2.4 Triangulation

A triangulation is made up of a collection of triangles on a set of points in a plane (Figure 2.15) and is a basic geometric data structure in computational geometry (Biniaz and Dastghaibifard, 2012). Triangulations are widely used in various applications for representing geometries such as terrain surfaces in GIS, modelling structures in particular in the automotive industry with computer aided design systems and deriving relations between geographical objects in GIS. Also, finite element methods in engineering widely use triangulation to simulate physical phenomena with mesh generation techniques (Hjelle and Dæhlen, 2006). For both theoretical and practical reasons, a particular triangulation must meet the following requirements (Hjelle and Dæhlen, 2006).

1. No triangle $t_{i,j,k}$ in a triangulation Δ is degenerate, that is, points P_i, P_j, P_k are not collinear.
2. The interiors of any triangle in Δ do not intersect, that is, if P_i, P_j, P_k and $P_\alpha, P_\beta, P_\gamma$ are points of two triangles, then $\text{Int}(t_{i,j,k}) \cap \text{Int}(t_{\alpha,\beta,\gamma}) = \Phi$.
3. The boundaries of two triangles in a triangulation can only intersect at a common edge or at a common vertex.
4. The union of all triangles in a triangulation Δ is equal to the domain over which the triangulation is defined, that is, $\Omega = \cup t_{i,j,k}$
5. The domain Ω must be connected.
6. The triangulation Δ shall not have holes.
7. If v_i is a vertex at the boundary of domain Ω , then there must be exactly two boundary edges that have v_i as the common vertex. This implies that the number of boundary vertices is equal to the number of boundary edges.

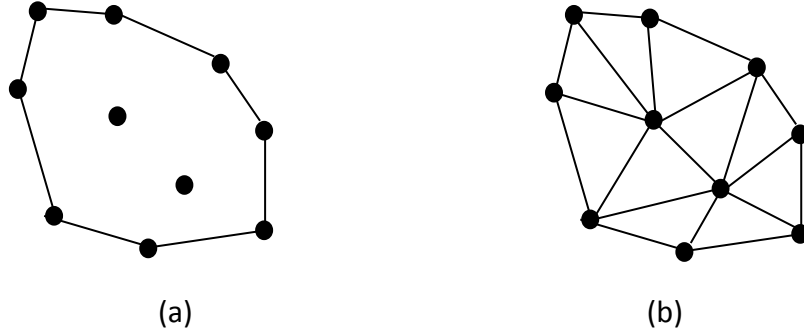


Figure 2.15 (a) A domain (Ω) with a set of points and (b) triangulation of the points.

2.4.1 Delaunay triangulation

It is observed that given a set of points P , their triangulation is not necessarily unique since a sequence of edge swaps could be applied to obtain a new valid triangulation. As a result, there can be poorly shaped triangles which are elongated or almost degenerate in some triangulations. This has given rise to the need to optimise triangulation of a set of points to create well-shaped triangles where the interior angles are almost close to equiangular. Among all possible triangulations on a set of points P , Delaunay triangulation named after Boris Delaunay (Delaunay, 1934) is the most optimised triangulation in terms of shape of triangles in that the triangulation maximises the minimum angle of its triangles (Hjelle and Dæhlen, 2006; Biniiaz and Dastghaibfard, 2012). When the triangulation is Delaunay, it has the property that no point in P is inside the circumcircle of any triangle in the triangulation (Berg *et al.*, 2008; Biniiaz and Dastghaibfard, 2012) (Figure 2.16(b)).

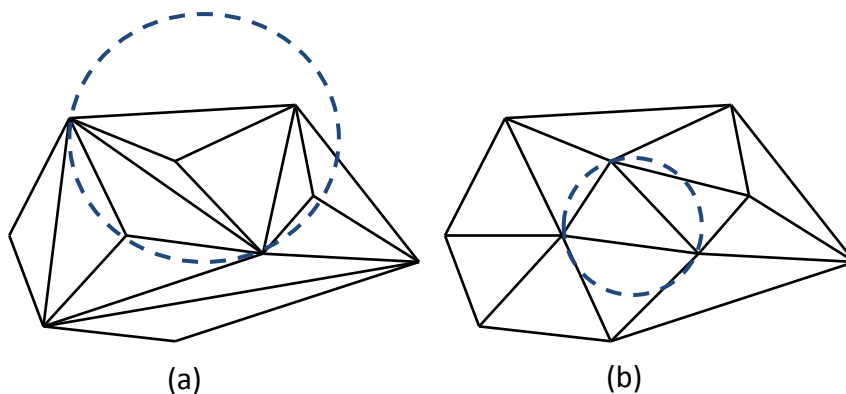


Figure 2.16 (a) Non-Delaunay and (b) Delaunay stable triangulation, based on Žalik (2005).

Biniáz and Dastghaibfard (2012) have categorised sequential algorithms for the construction of the Delaunay triangulation into five groups according to Su and Drysdale (1997). They are:

- Divide and conquer algorithms - these algorithms recursively partition the point set and apply local Delaunay triangulation in each partition. Next, on a merging phase resultant local triangulations are joined. The recursion stops when the size of the point set matches some given threshold.
- Incremental insertion algorithms - these algorithms insert points into the point data set: P one by one. The triangle containing the point inserted is further subdivided to form new triangles and then the circumcircle criterion is tested recursively on all triangles adjacent to the new ones and if required, their edges are flipped to meet the Delaunay property. Initially, it is easy to start with an auxiliary triangle such that all other points in P are inside it (Guibas, Knuth and Sharir, 1992; Kolingerová and Žalik, 2002; Žalik and Kolingerová, 2003).
- Gift-wrapping algorithms - the technique of these algorithms is that it first starts with a single Delaunay triangle and then incrementally discovers valid Delaunay triangles one at a time (Shewchuk, 1999). A similar kind of technique known as step by step approach is described in Hjelle and Dæhlen (2006).
- Convex hull based algorithms – First, in these algorithms the point set P is transformed into three-dimensional (3D) space and then the convex hull is computed. Finally, the Delaunay triangulation is created by projecting the resulting convex hull back into 2D space (Biniáz and Dastghaibfard, 2012).
- Plane sweep algorithms - this is a most famous technique of solving 2D geometric problems with a line called the sweep line moving along the X or the Y axis over a set of points (Cormen *et al.*, 2009). First, the points are sorted. Next the points are glided over the 2D plane and stopped at event points which are the points at which sweep line meets site points. According to the literature, the plane sweep algorithm has been employed with two geometric primitives: a line by Fortune (1987) and Žalik (2005) and a circle by Adam *et al.* (1997), and Biniáz and Dastghaibfard (2012).

2.4.2 Constrained Delaunay triangulation

The constrained Delaunay triangulation (CDT) is the generalization of conventional Delaunay triangulation in order to constrain a set of planar edges \mathbf{E} in a triangulation comprising of a set of planar points \mathbf{P} (Hjelle and D h len, 2006) including the end points of edges \mathbf{E} , thus \mathbf{E} is a subset of edges in the final triangulation $\Delta(\mathbf{G})$. Constrained edges may represent hydrographic features, roads, mountain ridges and building sides in cartography or linear features in finite element grids. In the CDT algorithm, the requirement is that the constrained edges must serve as the edges of the triangulation $\Delta(\mathbf{G})$. That is; the edges between points in \mathbf{P} must not intersect the interior of any constrained edge \mathbf{E} or any triangulated region such as a hole or a region outside the exterior boundary of $\Delta(\mathbf{G})$. With this requirement, the definition of conventional Delaunay triangulation can be modified to come out with a new circumcircle criterion definition which is according to Hjelle and D h len (2006) is “a constrained Delaunay triangulation $\Delta(\mathbf{G})$ of a planar Points and Edges $\mathbf{G}(\mathbf{P}, \mathbf{E})$ also known as planar straight line graph (PSLG) is a triangulation containing the edges \mathbf{E} such that the circumcircle of any triangle \mathbf{t} in $\Delta(\mathbf{G})$ contains no point of \mathbf{P} in its interior which is visible from all the three nodes of \mathbf{t} ” (Figure 2.17(d)).

Further, according to Shewchuk (1999), an edge or triangle is deemed to be constrained Delaunay if it satisfies the two conditions: (a) its interior does not intersect an input segment (edge) and (b) it has a circumcircle that encloses no vertex of the point set that is visible from the interior of the edge or the triangle. When considering the properties of the CDT, it is not Delaunay stable since it can violate the empty circumcircle criterion (see the hatched triangle in Figure 2.17(d) below) used in conventional Delaunay triangulation as described in Section 2.4.1. Therefore, the CDT is not necessarily a Delaunay triangulation as implied by its name. However, like Delaunay triangulation, the CDT can be constructed to be totally Delaunay by the flip algorithm where initially the point set which also include end points of all the edges to be constrained is triangulated with an arbitrary triangulation. Then each input segment (constrain segment) is forced into the triangulation by deleting the edges of the initial triangulation which cross the input segments. Further, re-triangulation is applied on either side of the two resulting polygons of each constraint segment. Finally, the

flip algorithm is applied to the provision that the constraint segments are not flipped (Shewchuk, 1999). This method finally generates a constrained triangulation which is Delaunay stable. A similar kind of approach to make the CDT Delaunay stable has been developed by Nam, Kiem and Nam (2009).

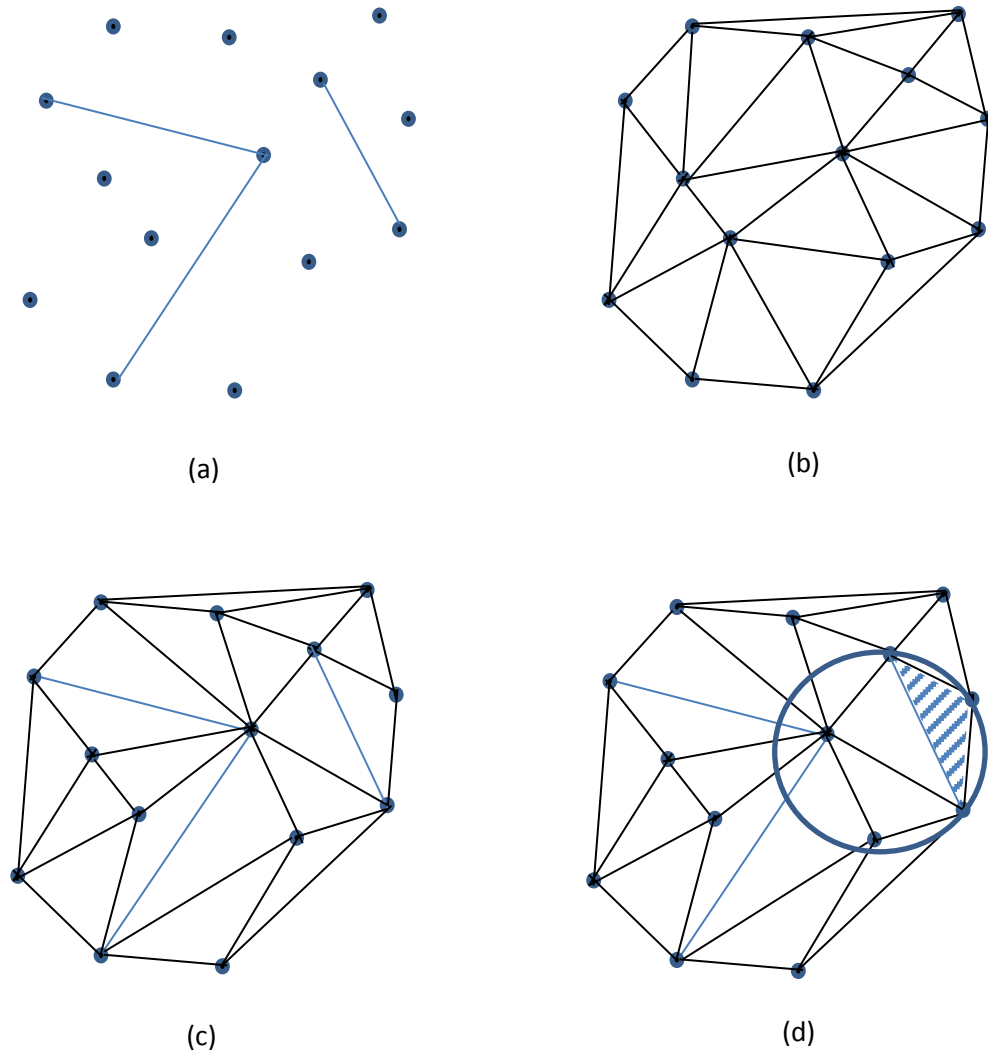


Figure 2.17 (a) A planar points and straight line edge graph $G(P, E)$ called PSLG (b) conventional Delaunay triangulation of point set P (c) CDT of $G(P, E)$ and (d) illustration of the modified circumcircle criterion on the hatched triangle for the CDT, based on Hjelle and Dæhlen (2006).

2.4.3 Conforming Delaunay triangulation

Triangulation of a planar straight line graph (PSLG) may form long and skinny triangles, leading to its numerical stability and convergence affected by the shape of the elements of PSLG in many applications. The approach to overcoming this problem is through the quality mesh generation as described by Bern and Eppstein (1992). Quality mesh generation offers techniques to guarantee the measure of the shape of triangles such as all non-obtuse triangles or all triangles with bounded aspect ratio which is the length of the largest edge divided by the length of the shortest altitude. According to Ruppert (1995), a simple measure of triangle shape is the minimum angle α , allowing a bound of maximum angle of $\pi - 2\alpha$, enabling a guarantee of an aspect ratio between $|1/\sin \alpha|$ and $|2/\sin 2\alpha|$. In order to obtain shape bounds to generate optimal triangle shapes, Steiner points have to be introduced which are not the original input vertices in the triangulation. According to Vreda et al. (2004), a Steiner point is a point with a particular geometric relation to a triangle although it is not part of the input set of points. Baker, Grosse and Rafferty (1988) have generated a triangular mesh where all angles are mostly at 90° with the smallest angle is at least 13° using a uniform square grid placed over the polygon to be triangulated with the grid spacing determined by the smallest object present (by the pair of closest vertices or by the closest pair of vertex-edge). Due to the mesh density being determined by the size of the smallest object, mesh density can be high resulting a large number of triangles in the mesh. Bern, Eppstein and Gilbert (1994) have introduced a mesh generation algorithm with both shape and size guarantee in replacement of a uniform grid (Baker, Grosse and Rafferty, 1988) by a quadtree which is a recursive subdivision into squares of different size yielding large triangles on large features. Further, Melissaratos and Souvaine (1992) have given some extensions to the quadtree algorithm to generate a Steiner triangulation with neither small angles nor obtuse triangles by introducing Steiner points. In Steiner triangulation, triangles are optimised with some criteria by adding Steiner points (Bern and Eppstein, 1992). A quite different technique of quality mesh generation is the Delaunay refinement approach where the properties of the Delaunay triangulation are preserved and maintained based on adding Steiner points following certain criteria. The Delaunay refinement algorithm presented by Chew (1989) triangulates a given polygon into a mesh in which all

the angles of the triangles are between 30^0 and 120^0 and all the edges are between $2h$ and h where h is a parameter chosen by the user to meet the angle criterion. Though their algorithm produces uniform mesh, there are cases where such a mesh has many more triangles than are necessary.

Conforming Delaunay triangulation (CNDT) is the process of creating a Delaunay stable mesh where triangles are generated using a set of Steiner points with some shape criteria inducing a Delaunay triangulation of the input PSLG. In this triangulation, each input edge must be a union of the edges of the Delaunay triangulation of the input vertices and the Steiner points (Bern and Eppstein, 1992). Ruppert (1995) has extended the work of Chew (1989) by introducing the CNDT algorithm to triangulate a PSLG such that all the angles of triangles between α and $\pi-2\alpha$ as described above where α is a parameter between 0^0 and 20^0 . In this approach, polygons, polygons with holes, objects consisting of multiple polygons, dangling edges and isolated vertices can be dealt with. This method is based on the Delaunay triangulation using two steps to maintain the mesh Delaunay stable by a refinement process: (a) find the Delaunay triangulation of all the input vertices of PSLG and (b) insert new vertices to split edges that need be served as constraint edges in the triangulation and to split a triangle with a vertex at its circumcentre to allow Delaunay stable mesh generation, sticking to the angle criterion in the Delaunay refinement process. Vertex insertion is mainly governed by two rules: (a) a segment is said to be encroached if a point lies within its diametral circle which is the (unique) smallest circle that contains the segment (vertex **p** lies inside the diametral circles drawn for segments **s3** and **s4** in Figure 2.18(d)) and any encroached segment that arises is immediately split by inserting a vertex at its midpoint and (b) split a triangle with a vertex at its circumcentre. The two resulting sub-segments have smaller diametral circles, hence may or may not be encroached themselves. See Figures 2.18 and 2.19 for the execution of the algorithm for clarity.

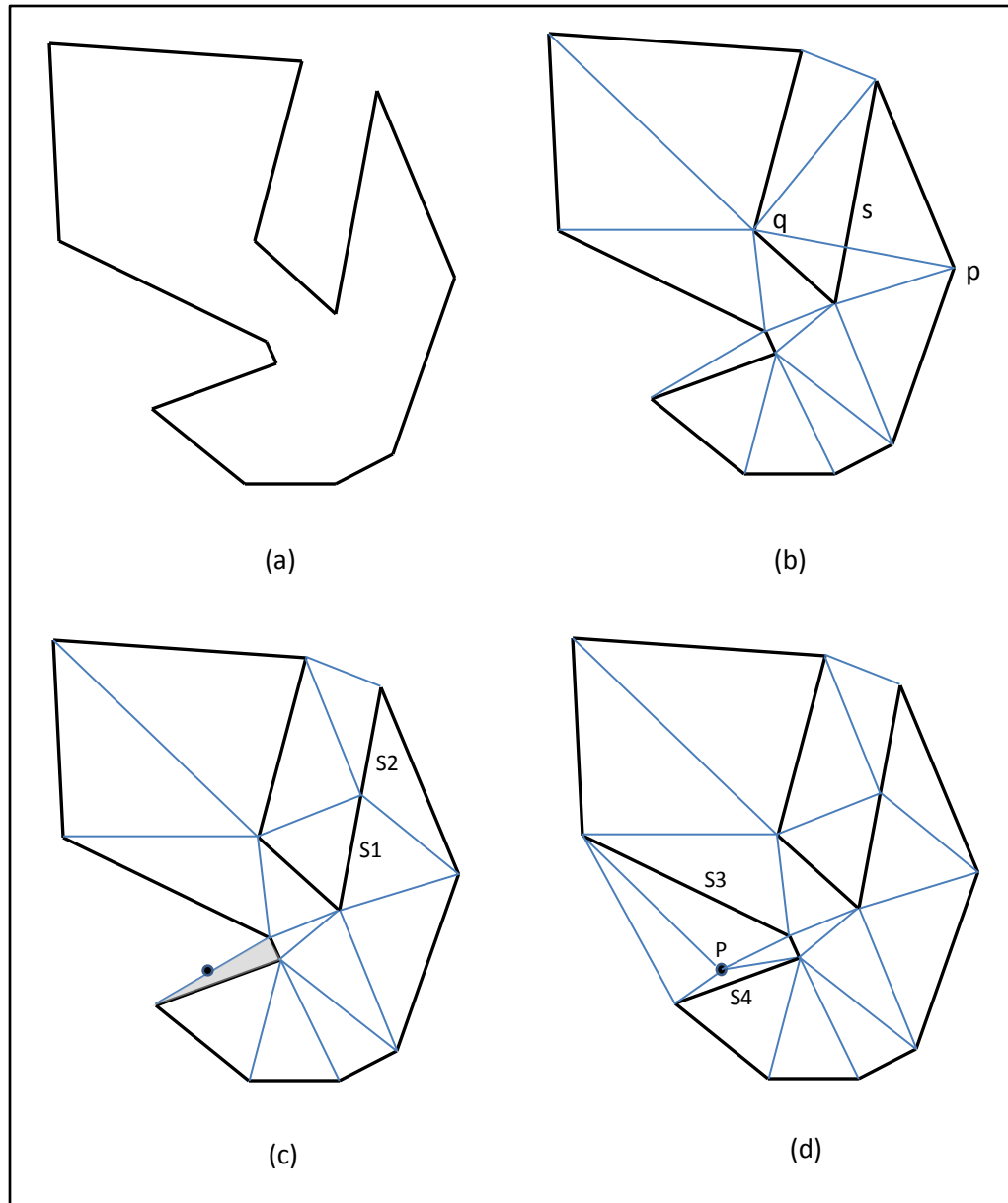


Figure 2.18 Execution of the CNDT algorithm on a simple example, based on Ruppert (1995): (a) input polygon (b) Delaunay triangulation of input vertices and constraint edge s is not a Delaunay edge since it crosses edge pq (c) segment s is split into $s1$ and $s2$ at its midpoint. Dot in (c) indicates its circumcentre of the shaded triangle with a minimum angle of 5.9 degrees and (d) circumcentre p is added to the triangulation where it encroaches upon segments $s3$ and $s4$.

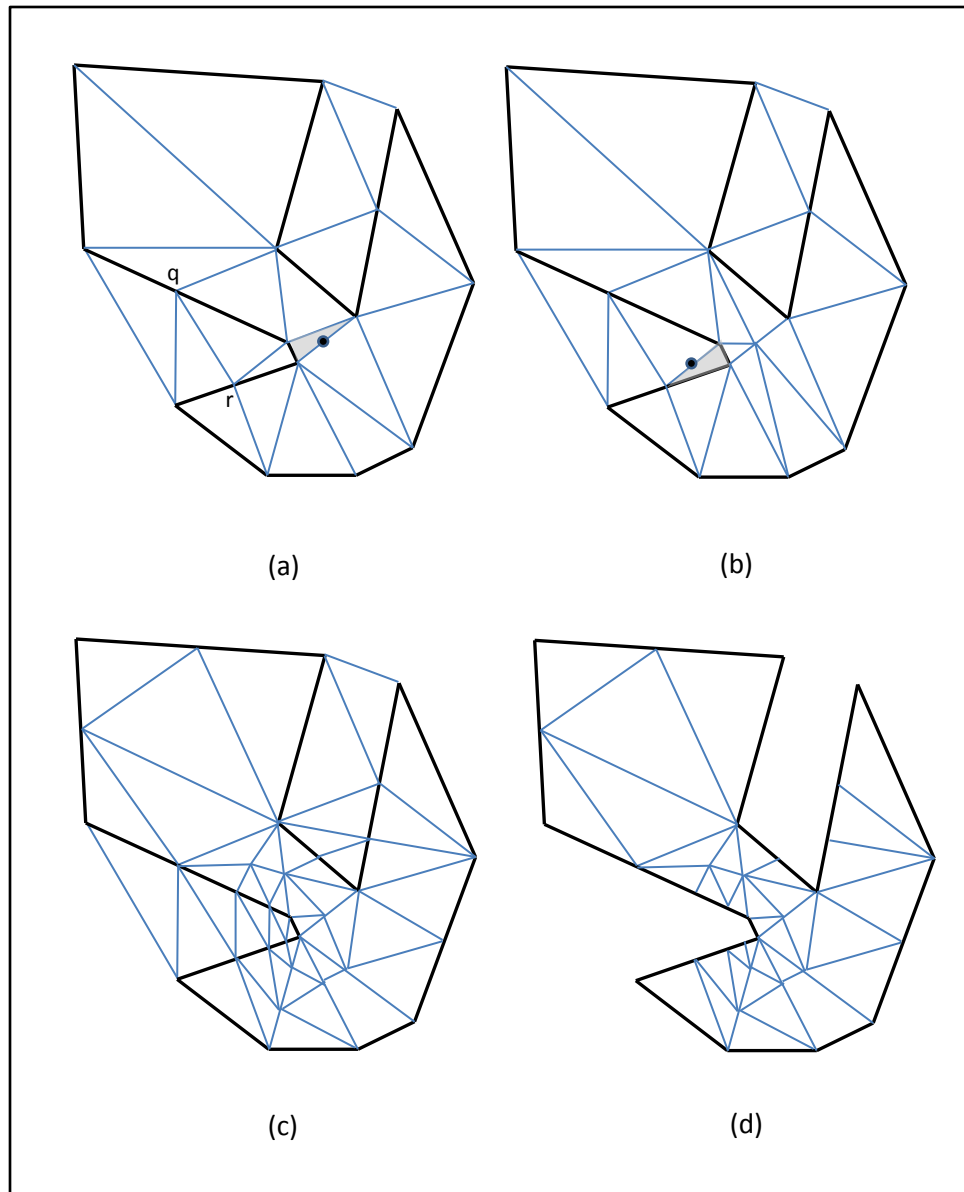


Figure 2.19 Continuation of the CNDT algorithm with splitting edges and triangles, based on Ruppert (1995): (a) two segments are split at q and r where shaded triangle now has a minimum angle of 9.8 degrees and will be split at the circumcenter (b) new minimum angle 11.6 degrees (c) allowing execution of split algorithm until the minimum angle becomes 25 degrees and (d) only triangles inside polygon is retained after removing external triangles.

2.4.4 Polygon triangulation

The decomposition of a simple polygon into triangles is polygon triangulation (Eberly, 2008). The notion of a simple polygon is that it does not have any vertex shared by more than two edges nor new vertices created by the intersection of two non-consecutive edges of the polygon. The simple polygon is also called a Jordan polygon according to Meisters (1975). That any triangulation of a simple polygon with n vertices should have $n-2$ triangles is a known fact in computational geometry. Various algorithms have been developed for triangulation of polygons. According to Meisters (1975), a polygon with four or more sides always has at least two non-overlapping ears where an ear is regarded as the region enclosed by the triangle V_1, V_2 and V_3 formed at the vertex V_2 if the chord joining V_1 and V_3 lies entirely within the Jordan polygon P comprising of consecutive vertices $V_1, V_2, V_3, \dots, V_n$ where $n \geq 4$. Ears (triangles) can be removed and stored for generating polygon triangulation using a recursive approach to triangulation. Improvement to Meisters (1975) algorithm has been done in terms of efficient implementation performance by Toussaint (1991). The implementation is based on searching for ears and cutting them off from the polygon recursively. Seidel (1991) has developed an algorithm for triangulating a simple polygon with the use of trapezoidal decomposition followed by the identification of monotone polygons. First, Chazelle (1991) has used trapezoidal decomposition, drawing horizontal chords from the vertices using the bottom-up approach and then applying the top-down approach to refine the triangulation based on polygon cutting theorem (Chazelle and Chazelle, 1982) and planar separator theorem (Lipton and Tarjan, 1979). Further, the polygon triangulation algorithm described by ElGindy, Everett and Toussaint (1993) divides the polygon P into two sub-polygons such that one of these sub-polygons is a good sub-polygon (GSP). A GSP is a polygon whose boundary differs from P in at most one edge. A proper ear of a GSP is an ear of P , and a GSP has at least one proper ear (ElGindy, Everett and Toussaint, 1993). Subsequently, the algorithm has been applied to finding a GSP and a vertex (a proper ear) recursively to retrieve all ears in the polygon P . However, one of the simplest and efficient algorithms for polygon triangulation, as mentioned by Eberly (2008), is ear-clipping. According to Eberly (2008), an ear of a polygon is a triangle formed by three

consecutive vertices V_1 , V_2 and V_3 of the polygon for which no other vertices are inside the triangle. In this context, vertex V_2 is called the ear tip (see Figure 2.20).

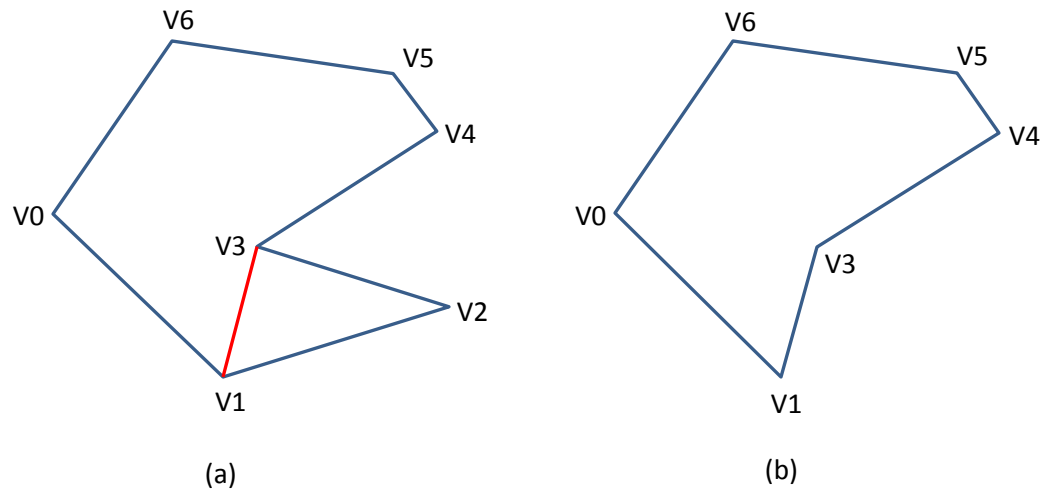


Figure 2.20 Recursive process of finding triangles in polygon triangulation: (a) A polygon with an ear (V_1 , V_2 , V_3) and (b) remaining polygon after removing the ear (V_1 , V_2 , V_3). Searching of ears is iterated until the remaining polygon becomes a triangle.

In the algorithm of Eberly (2008), if a polygon has $n \geq 4$ vertices, it locates an ear and removes (clips) it recursively as in the algorithm by Toussaint (1991) until a polygon with $n-1$ vertices is left with at the end of the process (a polygon with three ($n = 3$) vertices). Each ear thus clipped is a triangle in the final polygon triangulation process.

2.4.5 Related work on building geometries

In the data enrichment approach for building clustering with auxiliary data for subsequent generalization discussed in Section 2.3.3, the Delaunay triangulation spatial data structure has been the key technique used by Regnauld (2003, 2005), Basaraner and Selcuk (2004, 2008), Li *et al.* (2004), Ai *et al.* (2007), Regnauld and Revell (2007), Haowen, Weibel and Bisheng (2008) and Qi and Li (2008) because of its ability to derive adjacency relationships between contextual features that are to be dealt with in automatic map generalization. It should be noted that most of the GIS software and the spatial databases still lack the tools or flexible spatial data structures to derive auxiliary data. Adding auxiliary data also gives prior knowledge of the type of generalization operations to be applied based on the

characteristics of clusters of a data set. For example, clusters of buildings located very close to each other can be merged to form a single building entity at a smaller target scale while buildings located at the medium distance range can be represented by a reduced number of symbols usually called typification. Regnauld (2003, 2005), Regnauld and Revell (2007), Ai *et al.* (2007), and Qi and Li (2008) have used the CDT for the building clustering process. Further, it has been used by Jones, Bundy and Ware (1995) and Ware, Jones and Bundy (1995) for building and polygon aggregation and Ware and Jones (1996) for detecting and resolving spatial conflicts caused by polygons in map generalization. However, there is no guarantee that the CDT provides explicit neighbourhood relations between polygon objects such as buildings located in exceptional cases as shown in Figure 2.21. This issue has been discussed by Jones, Bundy and Ware (1995) and Ware and Jones (1996) citing a solution given by Jones *et al.* in 1995 using a search procedure that starts with the triangles externally connected to the object, and Ai *et al.* (2007). In order to overcome this exception, Ai *et al.* (2007) have first split the long edges (i.e. densification of edges) of building features into short segments before applying the CDT. However, the solution by triangulation based search procedure has not been used in the field of automatic map generalization. Another solution suggested for this exception is described in Section 2.4.2 above using the flip algorithm with re-triangulation. Further, none of the others have mentioned how they have worked on this issue in their applications in the field of building clustering and automatic map generalization.

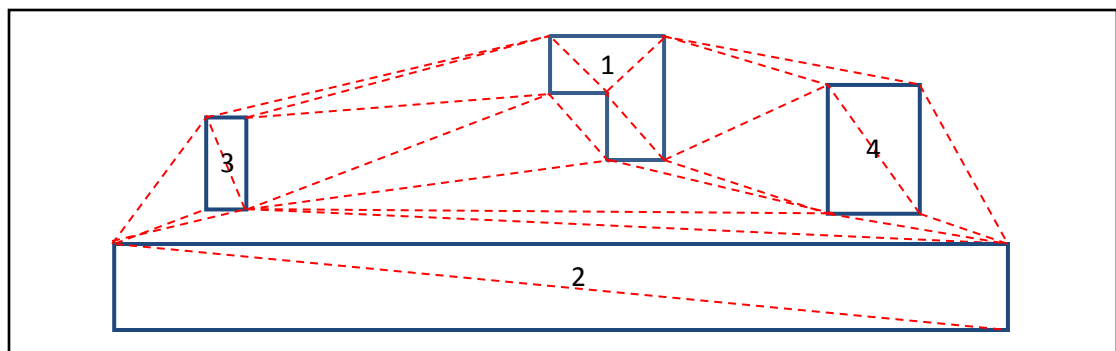


Figure 2.21 CDT of a set of building objects with constraining edges is shown in solid lines while other virtual edges are shown in dashed lines. The Figure shows that there is no object link connectivity between building with identification number (IDN) 1 and its nearest neighbouring building with IDN 2.

The type of constrained triangulation used by Li *et al.* (2004) and Haowen, Weibel and Bisheng (2008) on building objects has not been specifically mentioned by them. However, when observing the results of triangulation of both approaches, it is understood that they have either applied the CNDT or the CDT because intermediate vertices of building edges have also been triangulated. In the CNDT, these intermediate vertices are Steiner points that are not part of the source data as described in Section 2.4.3 above. In the case of the CDT, intermediate vertices can be added by densifying edges of buildings at appropriate, equal intervals before generating triangles as performed by Ai *et al.* (2007). Such densification splits the source input building edges into sub-segments affecting the source geometry. However, polygon triangulation, as described in the previous section, has not been applied in the fields of data enrichment and its subsequent automatic map generalization.

2.5 Knowledge discovery

With the amount of data, increasing in large volumes in databases day by day, the demand increases to extract useful knowledge from such large data sets. This requires intelligent and automatic analysis of such databases known as knowledge discovery in databases (KDDs). According to Frawley, Piatetsky-Shapiro and Matheus (1992), the term KDD is defined as “the non-trivial extraction of implicit, previously unknown and potentially useful information from data”. The KDD is an overall process of finding and interpreting patterns from data, while data mining only confines to data analysis (Elias, 2003) or narrowly defines as the application of computational, statistical or visual methods according to the literature (Mennis and Guo, 2009). Further, Witten, Frank and Hall (2011) have identified data mining as an automated technique for finding and describing structural patterns in data and as a tool for assisting in explaining the data and make predictions from them. According to Fayyad, Piatetsky-Shapiro and Smyth (1996), data mining is the application of specific algorithms for extracting patterns from data, identified as one of the steps in the KDD process. Further, the overall KDD process is iterative and involves multiple steps (Fayyad, Piatetsky-Shapiro and Smyth, 1996): (a) data selection (b) pre-processing - data cleaning (removal of noise) with deciding on strategies

to handle missing data, data filtering (finding the significance of data) and data transformation (harmonise different scale and attribute types) (c) incorporation of prior knowledge (enhancing auxiliary information of data to support analysis) (d) analysis carried out with computational algorithms and/or visual approaches (using data mining algorithms) and (e) interpretation of the results. The KDD process in relational databases deals with attribute types such as nominal (values as categories with no ranking), ordinal (values have a meaningful order), numeric values (values can be continuous, discrete and interval) and object relations. Further, automated knowledge discovery has become vital because spatial databases, which deal with data defined by location in addition to attribute data, have been rapidly growing in the applications such as geo-marketing, traffic-control and environmental studies (Ester, Kriegel and Sander, 1997) thus identifying spatial data mining as a growing research field in GIS applications (Miller and Han, 2009).

2.5.1 Data mining methods

Data mining methods are specific algorithms designed to predict or describe data. Prediction involves the use of some variables or fields to have unknown or future values of data (Fayyad, Piatetsky-Shapiro and Smyth, 1996). According to Fayyad, Piatetsky-Shapiro and Smyth (1996), existing data mining methods used to achieve goals in prediction and description of data are: (a) classification that maps (classifies) a data item into one of several predefined classes (b) regression that maps a data item to a real-valued prediction variable (c) clustering which seeks to identify a finite set of categories or groups to describe data (d) summarisation that deals with methods for finding a compact description for a subset of data (e) dependency modelling that deals with finding a model that describes significant dependencies between variables and (f) change and deviation detection that focusses on discovering the most significant changes in the data (a kind of sequence analysis).

These algorithms can be divided into two basic techniques called supervised learning (learning from examples) and unsupervised learning (learning from observations) according to the terminology of machine learning community (Elias, 2003). One of the

most popular supervised learning techniques used in spatial information retrieval (Sester, 2000b; Elias, 2003; Elias, Hampe and Sester, 2005) is the ID3 classification algorithm by Quinlan (1986). It is a symbolic learning and rule induction decision tree algorithm that learns by being told and looking at examples (looking at input/output matches of training data to find results for new data). It uses information gain as splitting criteria for decision tree formation based on a divide-and-conquer approach, working top-down, seeking at each stage an attribute to split on that best separates the classes and then recursively processing the sub-problems that result from the split (Witten, Frank and Hall, 2011). However, the ID3 algorithm can only deal with attributes with nominal values. It does not handle attributes with numeric values nor with missing values. Further, a series of improvements to ID3 culminated in a practical and influential system for decision tree induction called C4.5 (Quinlan, 1993). Such improvements include methods for dealing with numeric attributes, missing values and robust splitting criteria based on a gain ratio.

The unsupervised nature of the learning task leads to the classification over observations without the help of a teacher to pre-classify objects, but use an evaluation function to discover classes with a good conceptual description (Fisher, 1987), also known as clustering. Clustering is a common technique for statistical data analysis employed in many fields including machine learning, pattern recognition, image analysis, bioinformatics and information retrieval (Sharma, Bajapai and Litoriya, 2012). Among many clustering algorithms such as K-MEANS, DBSCAN, EM, CLOPE, OPTICS and CLASSIT, one of the incremental clustering algorithms attempted in the literature for spatial information retrieval (Elias, 2003; Elias, Hampe and Sester, 2005) is COBWEB developed by Fisher (1987), which is an incremental algorithm for hierarchical clustering based on the concept formation (Gennari, Langley and Fisher, 1989). The COBWEB algorithm does not split the instance space by testing a single 'teacher-selected' attribute at each node. A given instance belongs in the child node after incorporating it in the tree for which the category utility value has the highest probabilistic average. The category utility value is computed from all the attribute values based on a hill-climbing search where it identifies similarity of instances within the same class and dissimilarity of instances in different classes. The COBWEB tree is created with four main operators based on the employment

of the category utility value to determine which operator to employ at each decision point in the classification process: (a) classifying instance into an existing class (b) creating a new class (c) combining two classes into a single class (merging) and (d) dividing a class into a several classes (splitting) (Gennari, Langley and Fisher, 1989). One of the limitations in COBWEB is that it can handle only nominal attributes. However, CLASSIT is another hierarchical clustering algorithm further improved to work with numeric attribute values based on the COBWEB algorithm (Gennari, Langley and Fisher, 1989).

2.5.2 Salient landmarks

The landmarks based on the automatic map generalization can be used with different visualisation techniques as described in Section 2.1 in order to generate focus maps that give more emphasis in wayfinding. Further, as described in Section 1.1, salient landmarks help organise space because they stand out from their surroundings and serve as reference points in the environment so that they can support the identification of choice points where a navigational decision has to be made in wayfinding. Landmarks can be classified as local and global or on-route and off-route landmarks (Lovelace, Hegarty and Montello, 1999). On-route landmarks are positioned between nodes at decision points (a junction where a navigation decision is required) or at potential decision points (where a navigation decision is possible, but the route goes straight on). Off-route landmarks are directly neighboured to the route or at a distance like a tower or mountain chain. Accordingly, a landmark stands for a salient object in the environment and, therefore, indicates visual characteristic, unique purpose or meaning of a central or prominent location. Landmarks can further be divided into three categories considering these indicators: visual, cognitive and structural (Sorrows and Hirtle, 1999). According to Sorrows and Hirtle (1999), the more of these categories embed in a particular object, the more it qualifies as a landmark. Apart from landmarks, route information can be conveyed to the wayfinder in various presentation modes. For example, verbal instructions, although private and nonintrusive, can be problematic in public environments, and textual instructions on the navigation display can nevertheless require a high level of attention. As a result, humans tend to prefer to follow navigation instructions in a more natural way

with the use of landmarks along a route (Elias, Hampe and Sester, 2005). These landmarks can be represented graphically in a map-like depiction which provides a good overview knowledge, but sometimes can be difficult to interpret (Elias and Paelke, 2008). The research conducted by May *et al.* (2003) in order to find out information requirement for pedestrian navigation aids has shown that landmarks have been by far the most predominant navigation cue. The distance information and street names have been insignificant whereas the landmark information enables navigation decisions and enhances the pedestrian's confidence and trust. Raubal and Winter (2002) have specified properties to determine if a feature qualifies as an attractive landmark by taking into account its visual, semantic and structural characteristics. According to Nothegger, Winter and Raubal (2004), mapped routes enriched with landmarks at decision points lead to better guidance or less wayfinding errors than routes without landmarks.

2.5.3 Related work in deriving landmark saliency

In the application of knowledge discovery process, initially Sester (2000b) has employed the ID3 supervised learning classification algorithm for the interpretation of 2D map data. Then deriving salient landmarks for wayfinding has been carried out by Elias (2003) where all visual, structural and semantic characteristics that are required to determine a landmark to be salient have been obtained from a cadastral database for the subsequent use in data mining algorithms. In this approach, the ID3 and the COBWEB algorithms have been employed where COBWEB is an unsupervised clustering algorithm to identify the landmark saliency. According to Elias (2003), use of both algorithms has led to the desired results of identifying locally salient landmarks. However, ID3 has had the advantage of identifying discriminating attributes and their value domains directly over that of the COBWEB algorithm. Further, Elias, Hampe and Sester (2005) have used the modified ID3 algorithm to extract potential landmarks from a spatial database for the subsequent selection of route specific landmarks in a particular routing situation with the use of visibility analysis. However, the landmark saliency derived by Elias (2003) and Elias, Hampe and Sester (2005) has not been validated. Elias and Sester (2006) have introduced an approach to optimise route selection with point-like landmark buildings using qualitative

measures for the given potential set of landmarks chosen based on the approaches discussed by Elias (2003) and Elias and Brenner (2005).

Raubal and Winter (2002) have introduced a framework (formal model) to derive landmark saliency based on the statistical measures, considering visual, semantic and structural characteristics of landmarks. Then values of these measures for each property are integrated to assess local landmark saliency of a feature (i.e. hypothesis testing) at decision points by detecting outliers using the significance of deviations from the local characteristics of landmarks based on the mean and the standard deviation, assuming the data are normally distributed. However, this hypothesis has not been tested on a large data set to analyse the results, performance and computational cost. Nothegger, Winter and Raubal (2004) have derived landmark saliency for building facades at decision points and validated the results with a usability test involving human participation. The usability test has only taken into consideration the visual and semantic characteristics of the facades, neglecting structural characteristics. In this method, a significance score based on robust statistical measures to identify outliers using the median absolute deviation (MAD) of data is used to calculate the overall salience of a feature.

2.6 Problem scope

The CDT data structure has been used in many approaches to derive neighbourhood relations between polygon objects such as buildings in the field of automatic map generalization. However, deriving such relations between objects (especially buildings) located in exceptional cases as discussed in Section 2.4.5 has only been considered by Ai *et al.* (2007) by densifying edges of buildings in order to have triangle hooks between buildings to derive all possible neighbourhood relations. The disadvantage of this approach is that although additional vertices are added at regular intervals during densification of edges, this process is distance dependent, that is, depending on the densification distance, the result of triangulation varies with no guarantee that all possible neighbourhood relations are captured without a validation process. The use of the other solution mentioned for handling such exceptional cases by Ware, Jones and Bundy (1995)

and Ware and Jones (1996) using a triangulation based search procedure has not been applied in research related to map generalization. Several approaches (Jones, Bundy and Ware, 1995; Ware and Jones, 1996; Regnauld, 2003; Li *et al.*, 2004; Regnauld, 2005; Regnauld and Revell, 2007; Ai *et al.*, 2007; Revell, 2008; Haowen, Weibel and Bisheng, 2008; Qi and Li, 2008) have considered Delaunay triangulation between polygon features to derive neighbourhood relationships for automatic map generalization. However, none of the approaches have conducted an evaluation leading to the proper representation of neighbourhood links between polygon objects. Such an evaluation is required for further assessment of clustering and automatic map generalization to be adopted based on the CDT.

Although many approaches to clustering of building polygon features under data enrichment for subsequent generalization using the Delaunay triangulation have been adopted according to the literature (Regnauld, 2003, 2005; Li *et al.*, 2004; Revell, Regnauld and Thomas, 2005; Regnauld and Revell, 2007; Haowen, Weibel and Bisheng, 2008; Qi and Li, 2008), evaluation of clustering has not been tested and validated. Further, a systematic hierarchical clustering of building features, considering contextual features assisted by the Delaunay triangulation for proximity derivation together with a proper building orientation algorithm such as the statistical weighting by Duchêne *et al.* (2003) has not been developed and tested to support automatic map generalization of buildings.

The realisation of a robust building aggregation algorithm is understood to bring about a proper meaning to the aggregated buildings while preserving orthogonality of edges as much as possible as evidenced when investigating the results of some examples of such aggregation performed on building objects in the previous research by Li *et al.* (2004) and Haowen, Weibel and Bisheng (2008). Also, aggregation of building clusters containing irregular-shaped buildings with exceptional locations as described in Section 2.2.8 has not been dealt with in the previous work of building aggregation using both raster and vector techniques. Although the algorithm introduced by Regnauld and Revell (2007) creates orthogonal amalgams by aggregating rural building clusters, the results tend to become too simplified since all the buildings are represented with its MBR before aggregation.

Further, the MST to retrieve neighbouring pairs of buildings has been applied to the aggregation of buildings. It is not clear that the application of the MST alone would give proper neighbourhood relations for all the pairs of buildings considered in the aggregation.

There is a rich literature on spatial behaviour, spatial ability and cognitive aspects of the user in order to understand how people gain spatial knowledge with the assistance of cues in the real environment for various tasks (Lloyd, 1989; Gopal and Smith, 1990; Cornell, Sorenson and Mio, 2003; Li (2006); Brimicombe and Li, 2010). According to Kuipers (1978), people acquire spatial knowledge in positioning through route descriptions, topological relations of the road network and the orientation of objects in the environment. According to Elias and Paelke (2008), about 50% of all landmarks used in common wayfinding instructions are building features. Attempts made to visualise multiple representations of salient landmarks with automatic map generalization in the form of building objects by Elias, Hampe and Sester (2005) is one of the techniques towards generating focus maps for wayfinding. A hierarchy of multiple representations would also enable the human user to choose different types of building objects as landmarks at different representation levels, depending on the mode of navigation. However, automatic extraction of attributes required to extract salient landmarks subsequently from the spatial databases with data mining algorithms has not been extensively studied and realised, although the characteristics of landmarks to emphasise landmark saliency have been presented by Elias (2003) and Raubal and Winter (2002). Further, two data mining algorithms - ID3 and COBWEB - under KDD have been used (Elias, 2003; Elias, Hampe and Sester, 2005) in identifying salient landmarks for wayfinding. However, the results of the two algorithms have not been compared and validated with real data sets. Also, the output of landmark saliency obtained from these two algorithms has not been validated.

2.7 Research objective

The overall objective of this research is to investigate the processes and fill in the gaps of knowledge leading to the derivation of focus maps for wayfinding. The fields of research in this process are spatial data structuring with Delaunay triangulation, data enrichment, data mining techniques and automatic map generalization, focusing on building objects as salient landmarks.

2.8 Research questions

The following research questions will be addressed to fulfil the overall objective.

- How is the CDT data structure used and validated to derive explicit neighbourhood relationship between building polygons?
- How building objects are clustered with the hierarchical application of the Gestalt factors, considering contextual features with an intelligent cluster classification for the subsequent generalization process?
- Which generalization algorithms are the most influential for merging (aggregation) of building clusters depending on their outline shape (orthogonal or irregular) in order to provide a meaning to merging at coarser representation levels?
- Which data mining algorithm is the most appropriate to emphasise building landmark saliency?
- How are the focus maps validated in terms of the generalized results and derived landmark saliency?

2.9 Conclusion

This chapter introduces focus maps used for wayfinding and then investigates four important fields required to generate such maps with a review of related work in each field. Then it identifies the gaps to be addressed in each field in the process. Finally, it establishes the problem scope and formulates research questions to fill in the gaps in each field towards generating effective focus maps for wayfinding applications. The next chapter will discuss the methodology chosen to satisfy the information requirement to answer the research questions formulated in this chapter.

Chapter 3 Research Methodology

This chapter justifies the research design which is adopted in the research based on the previous studies in the four specific fields for generating focus maps identified through literature review in the previous chapter. It proposes methods to answer research questions to achieve the overall objective with a choice of existing tools, new tools and algorithms using open source software and algorithm libraries. Further, it discusses the approach to evaluating the results using the internal validation in each field with existing data sets chosen from the NMA of Sri Lanka and the Ordnance Survey (OS) of the United Kingdom. Finally, it describes the methods of external validation for the evaluating results of the focus maps generated using automatic map generalization and the identification of salient landmarks.

3.1 Related work for adopting the research design

The research design is selected by investigating the relevant testing and experiments already done in the four specific fields - triangulation data structure, data enrichment, automatic map generalization, and data mining - which are described below.

Triangulation data structure

Su and Drysdale (1995) have conducted an experimental comparison for computing Delaunay triangulation in terms of total runtime and high-level geometric primitives implemented using different triangulation algorithms on a number of non-uniform and uniform data distributions. Žalik and Kolingerová (2003) have tested a 2D incremental construction algorithm for Delaunay triangulation using the nearest point paradigm against time and worst case complexity. Further comparison has been made with some other existing triangulation algorithms using various distributions of input points representing both artificially generated data sets and the real point distributions. The constrained triangulation applied to building polygon data sets based on divide-and-conquer method using the principles described in Dwyer (1987) and Shewchuk (1996) has been compared with Delaunay triangulation in terms of runtime efficiency using data sets

with different details by Regnauld (2005). Erten and Üngör (2009) have also conducted an experiment to improve the performance of Delaunay refinement algorithms by developing two new algorithms and testing them against the refinement algorithm on the circumcircle insertion by Ruppert (1993) and off-centre insertion algorithm by Üngör in 2004 as mentioned by Erten and Üngör (2009) on many data sets with various properties such as complex boundaries with small angles and/or holes in the triangulation domain. The sweep-circle algorithm developed by Biniiaz and Dastghaibyard (2012) to compute the Delaunay triangulation has been tested and compared with the other popular Delaunay triangulation algorithms on runtime efficiency using various point distributions such as uniform, Gaussian, points arranged in clusters and points arranged in a tilted grid. In addition to the runtime, a test has been conducted to evaluate in-circle primitive of the algorithm with other algorithms.

Feature clustering under data enrichment

Zahn (1971) has discussed methods of clustering arbitrary point sets comprising of different cluster characteristics using the graph-based MST approach. This method is aimed at developing clustering algorithms based on examples from the 2D space to capture the human perception of Gestalts. Although this approach in automating clustering algorithms with a set of quantitative tools based on edge inconsistencies of MSTs is quantitative, experimental co-relation between the clusters of human perception and those determined by the quantitative tools has not been tested though suggested. However, Sadahiro (1997) has conducted an experiment to evaluate clustering results of a set of points obtained from clustering perception models with the manual clustering perception of subjects. In this experiment, subjects have been given a set of maps and asked to draw lines around points that they have perceived to be spatial clusters. Then the same set of maps has been digitised and processed through the GIS software for generating neighbourhood points for the development of cluster perception models. Finally, the results of cluster perception models have been compared with manual clustering perception results by the subjects in the validation process. Further, a hierarchical clustering algorithm implemented on a set of building polygon features by Qi

and Li (2008) has not been validated although future research on visual perception tests has been suggested.

Automatic map generalization

Bard (2003) has introduced an assessment model to evaluate generalization output with two levels of information: (a) information required to provide a generalization assessment (schema of geographic feature classes) and (b) information required to perform generalization assessment (i.e. algorithms for evaluation). The assessment model consists of mainly two features: (a) a characterisation function to report descriptive information about ungeneralized and generalized objects and (b) an evaluation function to report assessment information on the quality of generalization where evaluation corresponds to the computation and interpretation of the difference between the observed result and the theoretical result (of reference). The reference is defined by the values specified by the user and the initial characterisation of objects. Further, such values have been separated into two categories of specification: user specification and object specification. In this approach, quality information in the evaluation is divided into two: (a) detailed information (i.e. criteria by criteria) and (b) more general information (i.e. a value generated by aggregating all quality information of the features). In this method, the evaluation is coupled with visualisation of generalized output in zones where it gives unsatisfactory results. Bard (2004) has further explained the approach in detail with more theoretical background about evaluation functions, identifying a workflow of the generalization assessment. Stoter *et al.* (2009) have developed a methodology (framework) for evaluating automatic map generalization in commercial software with a broad study using four test cases of data, each from a different NMA, considering all sorts of map generalization problems. Their methodology mainly consists of two steps. In the first step, analysis of map requirements for automatic generalization with sourcing representative test cases, defining map specification in generalization constraints and harmonising constraints across the test cases has been carried out. The second step has been used to evaluate the generalized output using three integrated evaluation methods: (a) qualitative evaluation by the cartographic experts (b) automatic constraint-based evaluation and (c) evaluation which visually compares different outputs for one test case.

Upon the evaluation of the test cases it has been identified that the preservation constraints (e.g. on networks, patterns and spatial distributions) have been more difficult to formalise and evaluate automatically than that of the legibility constraints. Further experimentation on the evaluation of the preservation constraints on building pattern preservation in generalized outputs across different scales has been conducted by Zhang *et al.* (2012). The methodology has been implemented and validated on the interactively generalized data using the automatic pattern detection algorithms by Zhang, Ai and Stoter (2012) followed by data matching.

Deriving salient landmarks under data mining

Burnett, Smith and May (2001) have conducted an experiment to find out which landmarks are valued for vehicle navigation and their salient characteristics using two groups of subjects with three linked routes within an urban driving environment. On the analysis of data, a strong consensus on salient characteristics such as permanence, visibility, usefulness of location, uniqueness and brevity has emerged from their research. Raubal and Winter (2002) have used statistical measures for deriving landmark saliency on building features at decision points. However, the hypothesis has not been tested on a large data set as discussed in Section 2.5.3. Nothegger, Winter and Raubal (2004) too have derived landmark saliency based on the statistical measures. However, the results have been validated with building facades without considering the structural characteristics, using human participation in an urban area as discussed in Section 2.5.3.

3.2 Research design

Based on the related work in the previous section, the research design for emphasising landmarks for generating focus maps is experimentally based, and requires four phases of testing, evaluation, modification and validation of automatic processes in 2D spatial triangulation data structure, data enrichment, data mining and map generalization with the use of existing algorithms, modified existing algorithms and/or new algorithms to answer the research questions formulated in Chapter 2. In order to achieve this, open source algorithm libraries which are written in powerful Java object-oriented

programming language and the open source GIS software are used as the test environment. For spatial data storage and management, a powerful open source object-relational data management system called PostgreSQL coupled with the PostGIS spatial extension will be used. The reason to use such open source libraries and software is mainly because users have the ability to run, distribute, study and modify such libraries and software for any purpose. Open source is a collaborative software development platform that harnesses the power of peer review and transparency of the process to develop code that is freely accessible. Further, open source draws in a broad network of developers and customers from all over the world to drive innovation, unlike proprietary software. And it is worth to mention that the main data resource used in this research is the very rich and highly detailed OS MasterMap data at the scale of 1 : 1.25K owned and maintained by the OS which is the NMA of the United Kingdom. Further, topographic data at the scale of 1 : 1K from the NMA of Sri Lanka and some synthetic data will be used for the development and the testing of the algorithms.

3.3 Methods adopted

3.3.1 Constrained triangulation spatial data structure

In order to develop a proper constrained triangulation structure for polygon features which also applies to buildings and preserves the Delaunay property, existing constrained triangulation structures such as the CDT on sweep line algorithm by Domiter and Žalik (2008) and the CNDT on incremental algorithm by Ruppert (1995) will be implemented and compared with two modified constrained triangulation structures developed in this research based on the ear-clipping algorithm by ElGindy, Everett and Toussaint (1993) and the incremental point insertion based on the recursive edge-flipping algorithm by Berg *et al.* (2008) so as to investigate the efficiency and the effectiveness of retrieving explicit neighbourhood relations between polygon objects. The evaluation of the results of the algorithms will be carried out by developing prototypes at the implementation stage. Upon evaluation, the optimum algorithm out of the two new modified triangulation structures will further be validated including its runtime efficiency. A description of the

approach in detail together with resources and the type of data required for validation is as follows:

Resources:

- Synthetic data sets used for testing different algorithms.
- OS MasterMap digital data representing building polygons at the scale of 1 : 1.25K.
- Algorithm library of 2D spatial predicates and functions - Java Topology Suite (JTS) (Vivid Solutions JTS, 2013).
- Java object-oriented programming language.

Types of data required:

(a) For the validation of the modified constrained triangulation.

Primary data: Area of the convex hull of the triangulated region, total area of each triangle in the space region, number of triangles generated in the space region and the total area of polygons in the triangulated region.

(b) For testing the runtime efficiency of the modified constrained triangulation.

Primary data: Time taken to generate triangulation, number of triangles generated, number of neighbourhood links generated, number of buildings, number of building segments and the number of vertices in all buildings.

Data collection method(s):

Data collection will be carried out by the developed prototype with the necessary tools to extract these data from the four constrained triangulation structures explained above implemented with full automation techniques using object-oriented programming.

Data analysis:

Initially, a visual comparison between the results of the four constrained triangulation structures implemented will be carried out. Further, a quantitative evaluation will be

carried out in order to assess the modified constrained triangulation structure developed in terms of validity and runtime efficiency.

3.3.2 Spatial clustering of polygons under data enrichment

A hierarchical clustering algorithm will be developed based on the methodology of Qi and Li (2008) in the initial step to support subsequent generalization of building features with testing of different algorithms: (a) constrained triangulation developed in this research on incremental point insertion based on recursive edge-flipping algorithm by Berg *et al.* (2008) for generating explicit neighbourhood links (b) building orientation based on the MBR and the algorithm by Duchêne *et al.* (2003) and (c) the algorithm developed to find similarity in shape by Qi and Li (2008). The next step is to validate the clustering algorithm thus developed. For this purpose, an experiment which is similar to the visual perception test carried out by Sadahiro (1997) for validation of clusters on point features will be conducted by the NMA of Sri Lanka on the design developed in this research in which the evaluation of the algorithm consists of two phase data collection and analysis. The resources required and the methodology of the cluster perception test will be described in detail below.

Resources:

- Algorithm library of the 2D spatial predicates and functions: JTS (Vivid Solutions JTS, 2013).
- Algorithm library to perform conflation on spatial data sets: Java Conflation Suite (JCS) (Vivid Solutions JCS, 2013).
- Java object-oriented programming language.
- Hardcopy map representing building and road data at the scale of 1 : 4K in an urban area in Sri Lanka.
- Reduced hard copy, but not generalized, of the same data set at the scale of 1 : 10K.
- Digital map representing clusters around buildings together with road data at the scale of 1 : 4K from the NMA of Sri Lanka.

Types of data required:

Primary data: Graphics representing manual clusters, observation of subjects in figures and text to compare results of automatic clustering versus manual clustering. This data will be used from the NMA of Sri Lanka. Thus, such data become secondary in this work.

Data collection method(s):

Two phase focus group user testing will be carried out by the NMA of Sri Lanka. The second phase is based on the data collected during the first phase. In both phases, data collection is through participant observations. Two groups, each group consisting of fifteen (15) subjects, one group will consist of key informants who are cartographic experts in map generalization with the age in the range of 41-54 from the NMA of Sri Lanka and the other will consist of laymen from the clerical and management sections of the same NMA with the age in the range of 18-40, will be employed.

Data collection procedure: First phase

- i. Both groups are given a general idea of what clustering is about in automatic map generalization with some generalized outputs, explaining the polygon merging operator that will be used in the automatic map generalization process in this research.
- ii. They are given an idea (especially the lay group) of the map scale and how map scale will be used to determine ground distance between two objects based on the corresponding map distance between those two features depicted on the map.
- iii. The automatic clustering algorithm developed in this research is explained to both groups with the hierarchical constraints - proximity, orientation and similarity in shape - and the classification approach used for clustering. And more importantly, the purpose of this survey is to compare and evaluate the automatic clustering algorithm with manual clustering of each participant as to see how they perceive the application of hierarchical clustering on these three constraints.

- iv. They are given two hardcopy maps - one map at the scale of 1 : 4K comprising of building polygons and road network in an area of 0.75Km^2 ($1\text{Km} \times 0.75\text{Km}$) and the other map consists of the same data reduced and printed at the scale of 1 : 10K with no generalization applied. The participants are made aware of the metric scale printed on both maps so that the participants can read both maps and corresponding ground distance conveniently.
- v. Participants are instructed to apply manual clustering to buildings (see Appendix D.3) based on the three criteria used in automatic clustering in regions partitioned according to the road network in a systematic order as described by the researcher, starting from the North-West (top-left) followed by a zigzag route, ending at South-East (right-bottom) corner of the source map (1 : 4K) by investigating the target map at 1 : 10K to get an idea of probable clusters.

Data collection procedure: Second phase

The purpose of data collection at the second phase with the same subjects who participated in the first phase is to get a thorough evaluation of the results of the automatic clustering algorithm by comparing manual clustering results of each participant with the automatic results using a structured questionnaire with both open-ended and close-ended questions. Steps are taken to conduct the second survey before the elapse of one month of the first survey to avoid losing the impressions of the subjects involved in the first survey in the manual clustering process.

- i. In this phase, each subject in both groups is given the relevant manual clustering result on a map at the scale of 1 : 4K, a target map at the scale of 1 : 10K and a questionnaire to be filled by each participant to assess the quality of automatic clustering by comparing their own result with the automatic result.
- ii. The automatic clustering results are displayed to every subject in common on the screen. Subjects are allowed to request the researcher to zoom-in, zoom-out and pan areas of interest to view clusters on the digital map.

Data analysis: First phase

For the analysis of data in this phase, pre-processing of data is required to give meaning to the data collected. The following procedure will be adopted in this process.

- i. Results of manual clustering of each subject will be converted into digital format by assigning cluster labels manually. Then each cluster is polygonized with a convex hull formation. Therefore, at the end of the polygonization process, there will be thirty (30) digital maps containing polygon clusters.
- ii. Similarly, the results of automatic clusters will be polygonized by the same procedure.

Polygon clusters of the automatic results will be matched with the results of each participant of the entire source map region-wise to evaluate the results quantitatively after pre-processing. For this purpose, a prototype will be developed based on the work done by Revell and Antoine (2009) and a quantitative evaluation will be carried out based on the results of polygon matching.

Data analysis: Second phase

A qualitative evaluation will be carried out based on the data collected using the questionnaire to assess the quality of automatic clustering algorithms.

3.3.3 Automatic map generalization with building aggregation

Two new building aggregation operators, one is to aggregate clusters of orthogonal shape and the other is to aggregate clusters of non-orthogonal shape, will be developed to merge building polygons to be represented in coarse details as landmark polygons in this research. After the development, it is very important to evaluate the output of generalization results (Bard, 2003; Bard, 2004; Stoter *et al.*, 2009). The use of constraints is a common method of evaluating the automatic generalization process as discussed by Beard (1991), Sester (2000a), Barrault *et al.* (2001), and Ware, Jones and Thomas (2003). There are already two generalization assessment models based on constraints, each by Bard (2004), and Stoter *et al.* (2009) in the literature as discussed in Section 3.1.

Automatic evaluation by Bard (2004) has only assessed generalization output on 1 : 1 relation between the single type of building features. Further, the robustness of the algorithm has not been tested on a large number of features. However, the framework established by Stoter *et al.* (2009) for defining generalization specification and evaluating the generalized output using three methods: (a) qualitative evaluation by cartographic experts (b) automatic constrained based evaluation and (c) visual comparison of different outputs has been developed, considering all the aspects of problems in map generalization using comprehensive test cases chosen from the four reputed NMAs in the world. Therefore, for the evaluation of generalization output of this research, the framework of Stoter *et al.* (2009) will be used. Among the three methods of evaluation in the framework, the two methods leading to the evaluation of the generalized output using qualitative evaluation by cartographic experts and automatic constrained based evaluation are promising methods. However, considering the time and the cost involved, the third method which is based on a visual comparison of different outputs will be selected. For this purpose, the generalized output of this research will be visually compared with the output obtained from the industrial standard proprietary ArcGIS software (ArcGIS, 2013) by the Environmental System Research Institute (ESRI) and its methodology is described in detail below.

Resources:

- Synthetic data sets for developing and testing generalization algorithms.
- Digital source topographic data from the NMA of Sri Lanka, consisting of building and road features on the scale of 1 : 1K.
- OS MasterMap data at the scale of 1 : 1.25K.
- Generalization tools in the proprietary ArcGIS software.
- Algorithm library of the 2D spatial predicates and functions: JTS (Vivid Solutions JTS, 2013) and OpenCarto Java library (OpenCarto, 2013) for developing generalization tools in this research.

- Java object-oriented programming language.

The data used for both internal and external evaluations of the generalized output of the focus map is based on the two types of constraints: preservation constraints and legibility constraints. However, as encountered by Stoter *et al.* (2009), it is impossible visually to assess a threshold value used for a constraint. Therefore, these constraints are made such that they can be visually assessed with ordinal values - 'bad', 'fair' and 'well' - in the evaluation process.

Phase I: Internal validation of the testing algorithms

Types of data required:

Primary data in the form of text:

- (a) Preservation constraints: general orientation, general position, squareness, elongation and general shape.
- (b) Legibility constraints: area, dimensions (length/width of any part or edge) and granularity of edges.

Data collection method(s):

A prototype will be developed to visualize the generalized results using the algorithms developed in this research.

Data analysis:

A qualitative evaluation based on the visual comparison of the generalized results obtained from the test algorithms will be carried out.

Phase II: External validation of the generalized output in the focus maps

Types of data required:

It is the same as in phase I.

Data collection method(s):

Data will be collected using the generalization tools in the prototype developed to get the generalized results in the Phase I, and using the existing generalization tools in the ArcGIS proprietary software.

Data analysis:

A qualitative evaluation based on the visual comparison of the generalized results produced by the internally validated best algorithms in this research and the results generated from the generalization tools in the ArcGIS software will be carried out.

3.3.4 Emphasis of salient landmarks

This will be carried out in three phases in the research: (a) extracting all possible attributes required for subsequent knowledge discovery of salient landmarks from the spatial database automatically by developing a prototype (b) internal validation of the salient landmarks derived using three existing algorithms - ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) classification algorithms, and COBWEB (Fisher, 1987) clustering algorithm - with customisation in Java using the WEKA Application Programming Interfaces (APIs), comparing the output produced both at regions and decision points where navigational uncertainty would expect and (c) external validation of the results of salient landmarks in the focus maps generated in this research. For external validation of the salient landmarks, one of the two following methods are possible as discussed in Section 3.1: (a) an experiment with human participation (Burnett, Smith and May, 2001) and (b) using the two existing frameworks developed by Raubal and Winter (2002) and Nothegger, Winter and Raubal (2004). The first approach with human participation may be subjected to bias as mentioned by Burnett, Smith and May (2001). Further, in terms of cost, time and ethics, an experiment without human participation would be more convenient and experimentally controlled. Therefore, for the external validation of the output of salient landmarks in this research, the two frameworks by Raubal and Winter (2002) and Nothegger, Winter and Raubal (2004) will be implemented with a prototype development.

Resources:

- OS MasterMap data containing buildings and roads of two data sets representing the urban area.
- Open source PostgreSQL database with the PostGIS component to handle spatial data (PostGIS, 2013).
- Algorithm library of the 2D spatial predicates and functions: JTS (Vivid Solutions JTS, 2013).
- Open source GIS software - QGIS.
- Open source WEKA data mining algorithm library in Java developed by the University of Waikato (Weka, 2013).
- Java object-oriented programming language.

Phase I: Extracting visual, structural and semantic characteristics (attributes) of building objects required to identify landmark saliency

Types of data required:

Primary data: numbers and text - all possible attributes in terms of visual, semantic and spatial characteristics of building features that can be extracted from the available data sets.

Data collection method(s):

The data collection will be carried out by developing a prototype with necessary tools. Delaunay refinement incremental algorithm by (Ruppert, 1995) called the CNDT, and some PostGIS spatial functions will be used to extract structural properties. Semantics is directly obtained from the attribute information available in the spatial database while visual and structural characteristics will be derived from the existing tools and the new computational geometry algorithms developed in this research.

Data analysis:

This will be covered under the analysis of landmark saliency in the next phase (Phase II).

Phase II: Internal validation of the salient landmarks derived with COBWEB, ID3 and C4.5 data mining algorithms

Types of data required:

Primary data: Semantic, visual and structural properties of the salient landmarks derived in the Phase I above.

Data collection method(s):

A prototype will be developed to write the landmark saliency derived from the three data mining algorithms (COBWEB, ID3 and C4.5) in the PostGIS spatial database in which building features are stored.

Data analysis:

Both quantitative and qualitative evaluations will be carried out comparing the results derived from the three algorithms in regions covered by the contextual features and at decision points.

Phase III: External validation of the salient landmarks

Types of data required:

Primary data: visual, structural and semantic characteristics of buildings, salient landmarks derived from this research using the data mining algorithm evaluated to be the best during internal validation, salient landmarks derived from the two frameworks based on the values of significance, and Google street views of the identified decision points on the test data sets.

Data collection method(s):

A prototype will be developed to implement the two aforesaid existing frameworks to derive and compare landmark saliency with that of derived from the best data mining algorithm implemented in the prototype developed under Phase II. All derived results are written to the PostGIS spatial database by the new prototype.

Data analysis:

Both quantitative and qualitative evaluations will be carried out comparing the results with that of obtained from the two frameworks. A further cross-check of the results will be carried out with the Google street views.

3.4 Conclusion

This chapter presents and describes the methodology that are to be adopted in each of the four specific fields dealt with in this research to provide information requirement necessary to answer the research questions formulated in Chapter 2 to generate focus maps. Next chapter will describe in detail the implementation of tools required to develop spatial data structuring using object oriented Java programming for the subsequent use in data enrichment and automatic map generalization through rigorous testing and validation of existing algorithms, modified algorithms and/or new algorithms using both synthetic and real data sets.

Chapter 4 Implementation - I: Spatial Data Structure

This chapter presents how existing triangulation algorithms are tested and evaluated for further improvements and modifications to develop a constrained triangulation spatial data structure to handle polygon geometries specially with the enforcement of edge constraints, preserving Delaunay property and explicit neighbourhood relations. This data structure will subsequently be used for the data enrichment process required in automatic map generalization. It will also be used to develop the generalization algorithms required in generating focus maps in this research. The main reason for such testing and evaluation of a triangulation data structure is that the CDT structure used in many data enrichment and automatic map generalization applications, as described in Section 2.4.5, cannot generate explicit neighbourhood relations between polygons. Although a couple of solutions have been suggested to make the neighbourhood relations explicit: (a) a triangle based search procedure by Jones, Bundy and Ware (1995) and Ware and Jones (1996) and (b) a method of arbitrary triangulation initially applied on all points and further application of re-triangulation in areas, deleting initial triangles crossing constraint edges by Shewchuk (1999), none of the solutions have been tested and implemented in research related to the map generalization. Another reason is that when reviewing the literature, the CNDT data structure described in Section 2.4.3 has not been used and tested abundantly in applications related to data enrichment and automatic map generalization. The work by Bader and Weibel (1997) in producing polygon maps using automatic map generalization is among one of the few applications of the CNDT on polygon geometries.

For testing and evaluation (internal validation) of the triangulation data structures developed in this research, both synthetic and real data sets will be used. The testing platform is proprietary Visual C# and open source Java object-oriented programming languages with data stored both in ASCII and PostGIS formats with PostGIS working as a spatial extension to PostgreSQL object-relational database management system to handle spatial data.

4.1 Testing of an existing constrained Delaunay triangulation algorithm

The CDT using the sweep line algorithm by (Domiter and Žalik (2008)) is tested on both synthetic and real data. This algorithm is already implemented in the open source Poly2Tri CDT library (poly2Tri, 2012). However, the source code is modified in Java in this research to get the adjacency link between each pair of buildings with the use of unique building identification numbers (IDNs) as there is no provision to derive adjacency relationships between the triangulated polygons in the poly2Tri implementation. The reason is that no attribute is assigned for polygon objects to track them with the connected triangle edges in its implementation.

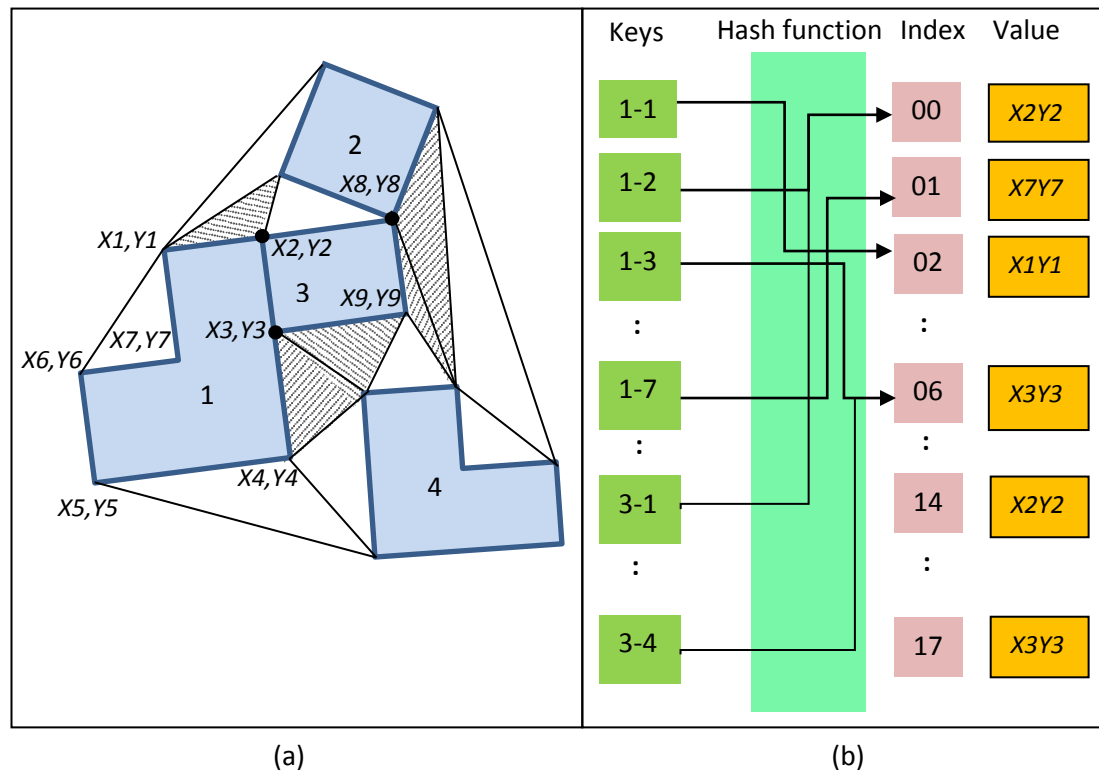


Figure 4.1 (a) CDT with duplicate nodes of triangles hatched in grey colour and (b) Hashtable data structure to handle duplicates where concatenated X and Y coordinate pair of each vertex is stored as a value against unique key consisting of a combination of building IDN and the vertex number (e.g. Key 1-2 denotes vertex 2 of building IDN 1) of each building polygon. Building polygon IDN 1 has seven vertices starting from $(X1, Y1)$ to $(X7, Y7)$ and building polygon IDN 3 has four vertices - $(X2, Y2)$, $(X8, Y8)$, $(X9, Y9)$ and $(X3, Y3)$ as shown in (a) above.

In this approach a unique IDN created for each vertex of all the polygons with the combination of polygon IDN and its vertex IDN is mapped with its concatenated pair of coordinate values (x, y) using the Hashtable data structure widely used in computer algorithms (Cormen *et al.*, 2009) in Java (Figure 4.1). This enables enriching each triangle edge vertex with the connected polygon IDNs by retrieving the concatenated pair of edge vertex coordinates from the Hashtable. A single vertex in a triangle can have multiple polygon IDNs in this instance as shown in Figure 4.2. Thus, from the polygon IDNs enriched at all three points of a triangle, adjacency relations of polygons can be derived. The adjacent links thus derived are automatically written to an ASCII file once triangulation is executed (see Appendix B.1 for poly2Tri user interface and an example of adjacency links with constrained Delaunay triangulation on a synthetic data set).

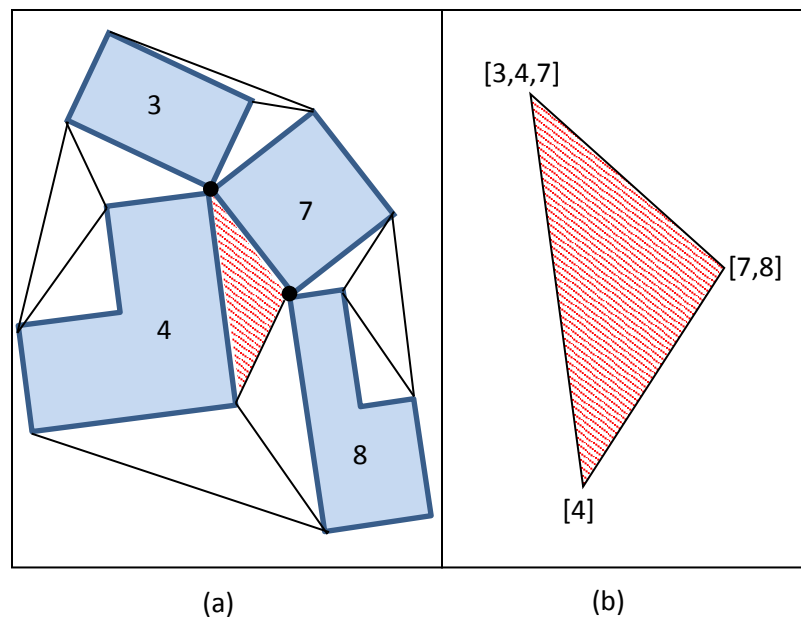


Figure 4.2 (a) CDT with a triangle comprising of duplicate nodes hatched in red colour and (b) the same hatched triangle with duplicate node IDNs generated from building number attached at three corners. The attached building IDNs in the hatched triangle derive adjacency links (3,4), (4,7) and (4,8) between buildings with IDNs 3, 4, 7 and 8.

4.1.1 Input data structure

The input data file for the building configuration depicted in Figure 4.3(a) should have the ASCII format given in Figure 4.3(b). This file is created automatically from the building geometries stored in PostGIS with a prototype developed in Visual C# 2008 (see Appendix A.1) to read data from a PostGIS query (see Appendix E.1). In creating this file, building geometries which are the features to be triangulated must be enclosed in an outer bounding polygon. The coordinates of this outer polygon are stored counter-clockwise while the coordinates of the inner polygons are stored clockwise in order to differentiate bounding polygon from all other inner polygons by the algorithm. The prototype uses minimum bounding box (MBB) of all building geometries to create the outer bounding polygon with a small outward offset from the MBB. The reason to use a small outward offset is that this algorithm does not work if the outer bounding polygon touches one or more inner polygons. Further, the IDN of outer polygon is assigned minus one (- 1) while the other inner polygons are assigned random or serial unique IDNs in order to differentiate outer bounding polygon from all other inner polygons in the prototype. In the ASCII file the number of vertices of the outer polygon and each inner polygon should be on top of each polygon information (building polygon IDN and its coordinates of each vertex) in the file as shown in the Figure 4.3(b).

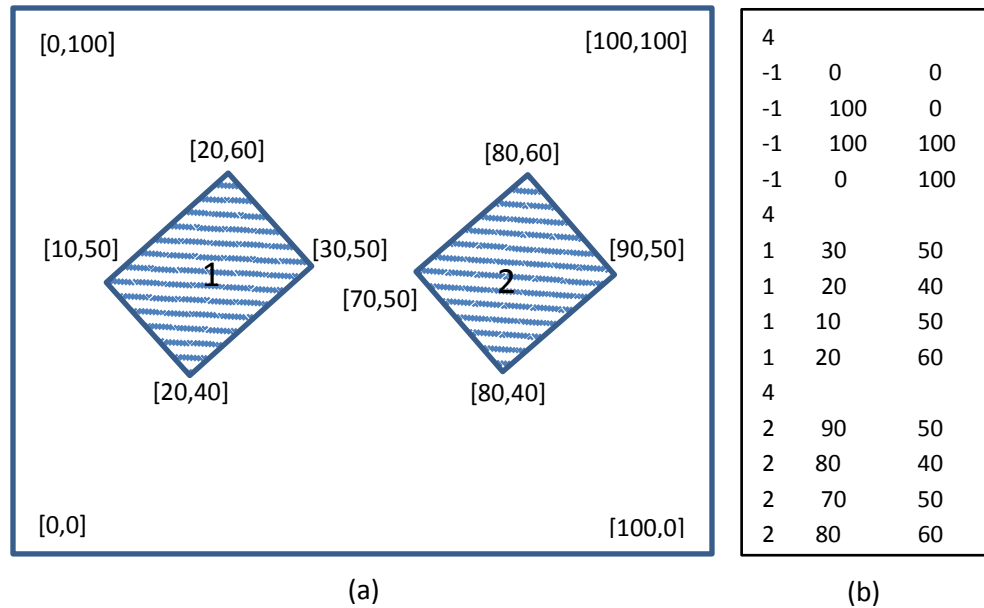
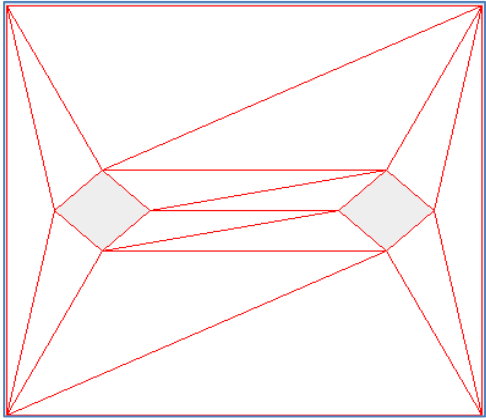
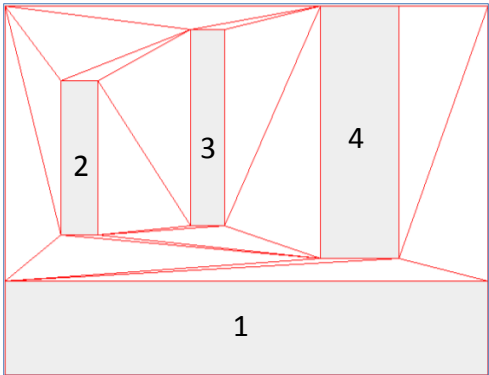
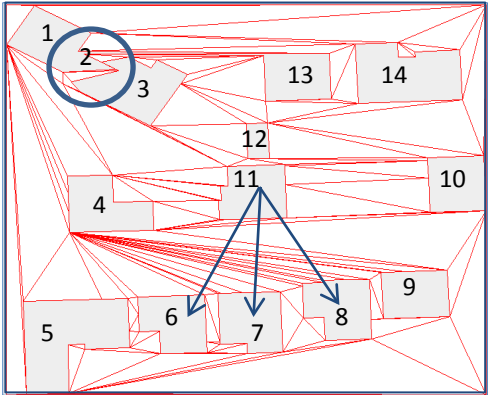


Figure 4.3 (a) Outer polygon and inner building polygons (triangulating features) with corner coordinates and (b) representation of the outer polygon and the two inner building polygons in ASCII format.

4.1.2 Evaluation of the results of constrained Delaunay triangulation

The algorithm by Domiter and Žalik (2008) does not weaken the Delaunay property described in Section 2.4.1 as they adapt re-triangulating regions formed by the deletion of triangles that are intersected by constrained edges, described in Section 2.4.2, as discussed by Shewchuk (1999). However, the Delaunay property is not stable in the implementation of the CDT in open source Poly2Tri library despite it being mentioned that the triangulation is based on the work of Domiter and Žalik (2008) when considering the results #2 and #3 in Table 4.1 - result #2 does not represent explicit neighbourhood relations (no connection between building IDNs 1 and 3) and result #3 produces both skinny triangles and implicit neighbourhood relations (building IDN 11 in the middle has no connection with building IDNs 6, 7 and 8 shown with arrows). It is also found that the triangulation cannot handle polygons that share common boundaries when observing the three adjoining building IDNs 1, 2 and 3 circled in the top left corner of result #3 in Table 4.1 (vertices of building IDN 2 have been snapped to incorrect positions, distorting the shape of the three attached buildings). When observing the results, it can be concluded that the CDT does not provide explicit adjacency relationships between polygon geometries.

Table 4.1 Results of the CDT based on the sweep line algorithm by Domiter and Žalik (2008) with different data sets.

#	Type of data	Results of the CDT
1	Synthetic data set represented in Figure 4.1(a)	
2	Synthetic data set comprising of narrow buildings with long and parallel edges	
3	OS MasterMap source data (1 : 1.25K) comprising of edges of all buildings with intermediate vertices except buildings 2, 12 and 14	

4.2 Testing of an existing conforming Delaunay triangulation algorithm

The CNDT class which is available in JTS 2D spatial algorithm library based on the incremental algorithm by Ruppert (1995) is implemented in a prototype (Appendix B.2) for this research and tested on the same three data sets used in the previous Section 4.1.

4.2.1 Input data structure

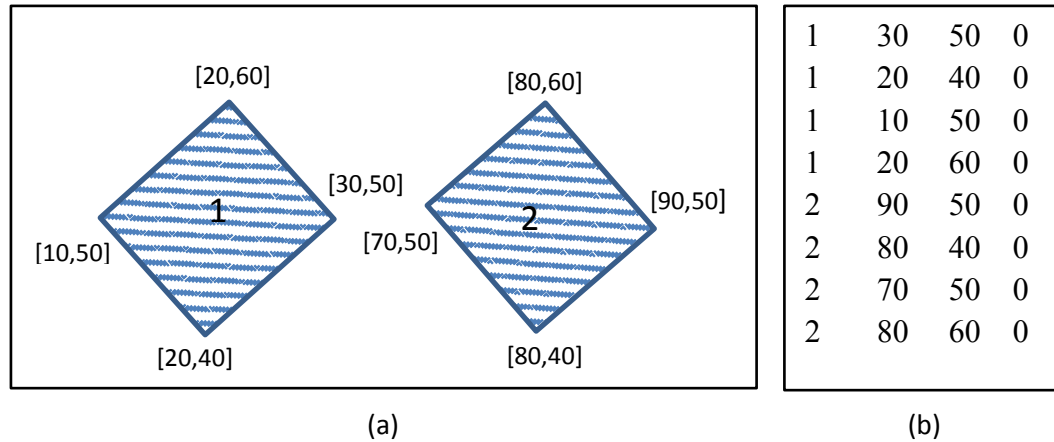


Figure 4.4 (a) Building polygons (triangulating features) with corner coordinates and (b) representation of two building polygons in ASCII format.

Building polygon data in ASCII space delimited format with the 3D coordinates (height - Z is zero) of vertices of each polygon stored either counter-clockwise or clockwise, can be used to generate triangulation with the prototype built (Appendix A.2). In this case, only the building outlines are dealt with. Feature IDN of each polygon should be coupled with the coordinates of each vertex as depicted in Figure 4.4(b). Further, this prototype can extract each line segment with its IDN and respective coordinates similar to the representation of building polygon data in ASCII format. The link between each polygon, between each line and polygon, and between each line is tracked by the unique coordinate of each source polygon/line coordinate mapped with polygon/line IDN as explained in Section 4.1 for deriving neighbourhood relations. The incorporation of both the polygon and the line geometries has been facilitated in generating the ASCII data file since the CNDT implementation in JTS can also generate triangulation between line geometries.

4.2.2 Enriching Steiner points

The Steiner points as explained in Section 2.4.3 generated to make the triangles Delaunay stable in the triangulation process are foreign vertices that are not in the source data set. Therefore, to identify which edge of the object is split virtually and linked to the triangles thus generated, the following algorithm is developed and adopted in this research as no members (attributes) are set to link the triangulation between polygons or between polygons and lines with their corresponding triangle edges in the CNDT class in JTS.

- i. Iterate the list of triangles generated by the use of triangulation class and get the first triangle.
- ii. Check from the mapped coordinate against IDN of each vertex of an object, if an edge of the triangle is linked to an object.
- iii. If it is not linked to an object, retrieve all objects that are close to the triangle using spatial indexing based on the sort-tile-recursive (STR) algorithm by Leutenegger, Lopez and Edgington (1997) implemented in JTS. R-Tree is a dynamic index structure widely used in spatial databases to retrieve data objects of non-zero size quickly, particularly polygon geometries of multi-dimensional spaces according to their locations (Guttman, 1984).
- iv. Create a small buffer with a small tolerance around the vertex of the triangle edge that is not linked to an object.
- v. Get all objects that intersect this buffer and link their IDNs to the triangle vertex.
- vi. Check two other edges of the triangle and adapt the steps (iii) to (v) if objects are not linked to the edges.
- vii. Go to step (i) to choose the next triangle until the end of the triangle list.

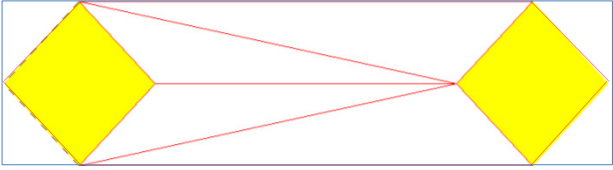
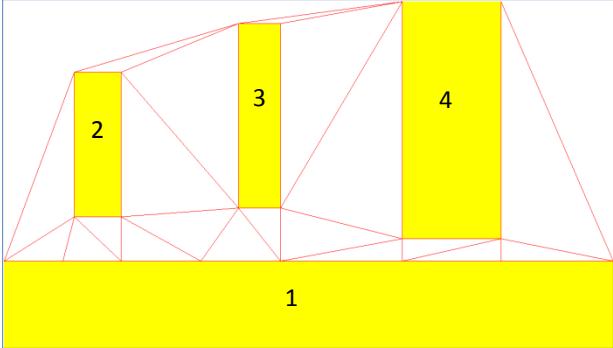
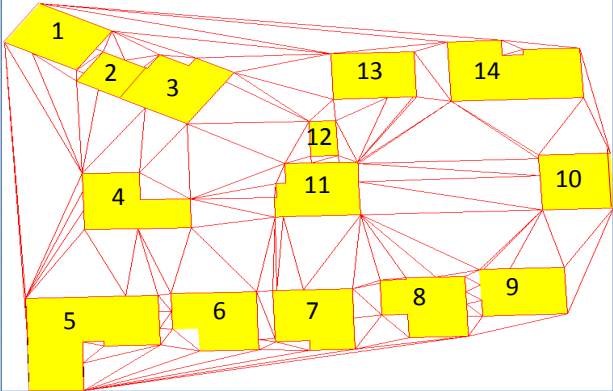
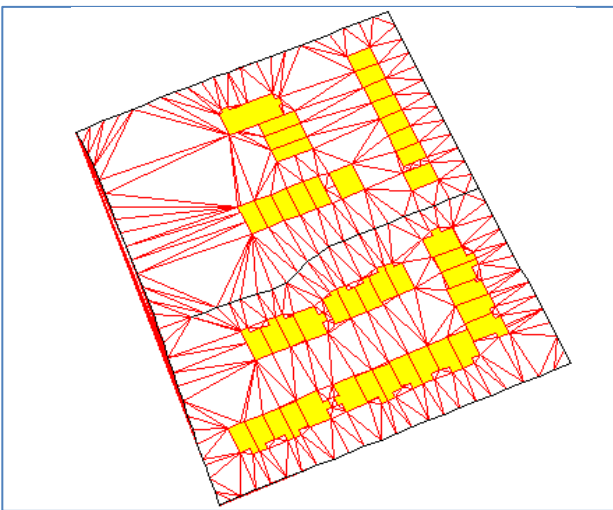
4.2.3 Evaluation of the results of conforming Delaunay triangulation

When investigating the results #2, #3 and #4 in Table 4.2, the triangulation represents explicit neighbourhood relations between the buildings, and between buildings and roads. It can also handle buildings that share a common edge (building IDNs 3 and 4 in the result #3 and most of the buildings in the result #4). However, the CNDT splits lines and edges of the polygons virtually introducing new vertices (Steiner points) thereby allowing more triangle hooks between polygons, between lines and polygons, and between lines in the result as depicted in Table 4.2 (compare triangles between building IDNs 1 and 2, 3 and 4 in the result #2 and triangles between building IDNs 5 and 6, 7 and 8, 8 and 9, and 11 and 12 in the result #3 in Table 4.2 with triangles generated using the CDT on the same building data sets in Table 4.1 for the identification of Steiner points) in order to make the triangulation Delaunay stable as explained in Section 2.4.3.

Since more hooks are added to the triangulation, this type of triangulation is useful in aggregating building polygons with least exaggeration in bridging the gaps between such polygons. However, exceptions have to be dealt with when triangles consisting of Steiner points are handled with source polygons in geometrical operations such as aggregation (union) since Steiner points thus added are not part of the segments of the original polygons (edges of polygons are not split into the sub-segments). Further, this triangulation structure can be utilised to derive all possible adjacency relationships between polygons, between lines and polygons, and between lines with the help of more hooks, along with the algorithm developed in this research given in Section 4.2.2.

There is a need to improve the CDT algorithm to handle explicit neighbourhood relations so that it generates Delaunay stable triangles which include only the source points in the polygon data set. Such a triangulation can be utilised not only in deriving neighbourhood relations in the data enrichment process, but also in the development of generalization algorithms such as polygon aggregation, depending on the application requirement.

Table 4.2 Results of the CNDT based on the incremental algorithm by Ruppert (1995).

#	Type of data	Results of the CNDT
1	Synthetic data set represented in Figure 4.1(a)	
2	Synthetic data set comprising of narrow buildings with long and parallel edges	
3	OS MasterMap Data (1 : 1.25K) comprising of edges of all buildings with irregular splits except buildings 2, 12 and 14. However, new Steinar points are introduced between buildings 5 & 6, 7 & 8, 8 & 9 and 11 & 12	
4	OS MasterMap Data (1 : 1.25K) consisting of both lines (roads) and polygon (buildings) geometries	

4.3 Testing of a new constrained algorithm on polygon triangulation

When considering the results of both CDT and CNDT implementations using Poly2Tri and JTS open source Java libraries, it is understood that the CDT which is completely Delaunay unstable with skinny triangles, will have triangle edges connecting only existing polygon vertices with implicit neighbourhood relations while the CNDT implementation with JTS open source Java libraries as described in Section 4.2.3 has produced Delaunay stable triangles with explicit neighbourhood relations with the addition of new vertices (Steiner points). The new algorithm for constrained triangulation will use the open source Java source code of the polygon triangulation algorithm on classic ear-clipping by Eberly (2008) discussed in Section 2.4.4, available for download in the JTS forum (JTS Topology Suite, 2012). This triangulation will also have triangle edges only connecting existing polygon vertices (compare the results #2 and #3 in Table 4.3 with triangles generated using the CNDT on the same building data sets in Table 4.2).

4.3.1 Input data structure

The input data structure is the same as that of the CNDT described in Section 4.2.1. The same prototype, as given in Appendix A.2, is used for input data file creation. Further, the triangulation can be processed by reading polygons from PostGIS directly as well. For deriving neighbourhood relations, the same approach with mapping polygon IDNs with polygon vertices explained in Section 4.1 is used. The results of the triangulation on different data sets are given in Table 4.3.

4.3.2 Triangulation algorithm

A new constrained algorithm for polygon triangulation is implemented in the same prototype mentioned in Section 4.2 (Appendix B.2) and described in Figure 4.5.

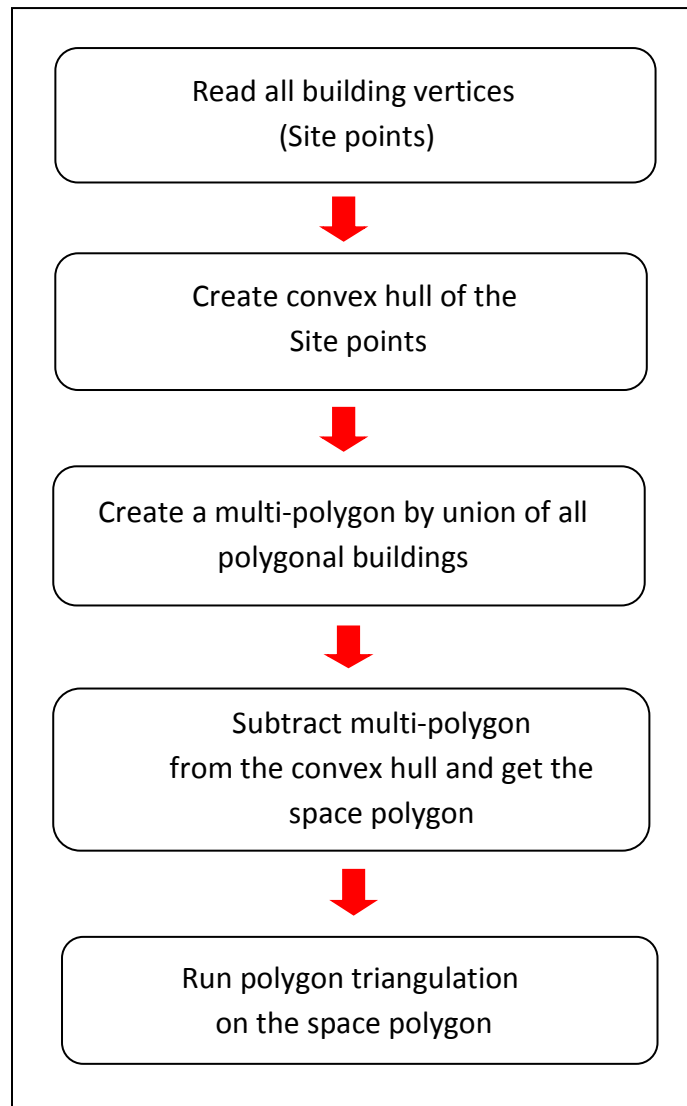
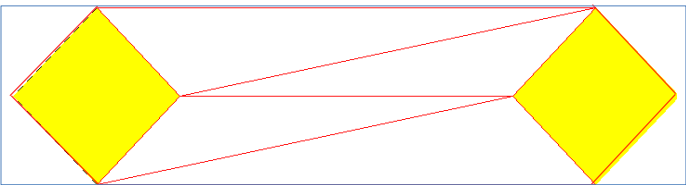
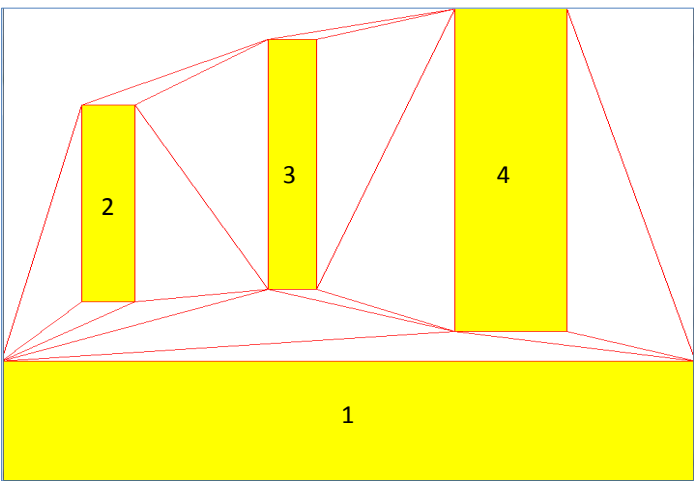
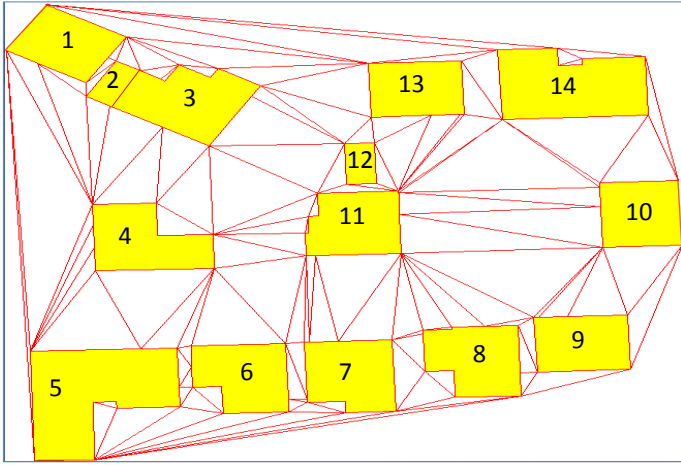


Figure 4.5 Constrained triangulation algorithm based on the polygon triangulation on ear-clipping by Eberly (2008).

The results of the implemented constrained triangulation algorithm on the data sets used for testing are given in Table 4.3.

Table 4.3 Results of the constrained triangulation developed in this research based on the polygon triangulation algorithm by Eberly (2008) on the source data.

#	Type of data	Results of the constrained triangulation
1	Synthetic data set represented in Figure 4.1(a)	
2	Synthetic data set comprising of narrow buildings with long and parallel edges	
3	OS MasterMap source data (1 : 1.25K) comprising of edges of all buildings with intermediate vertices except buildings 2, 12 and 14	

4.3.3 Evaluation of the results of constrained algorithm on polygon triangulation

When considering the results #2 and #3 in Table 4.3, the triangulation represents the explicit neighbourhood relations between the buildings. It can also handle buildings that share a common edge (building IDNs 2 and 3 in the result #3). Further, the triangulation does not split building edges, inserting new vertices in generating triangles. This is an advantage because this triangulation can generate explicit neighbourhood relations in generating well-shaped triangles by using the input source data set (no Steiner points added), comparing to the skinny triangles generated in the result #3 of Table 4.1 with the same data set used in the CDT described in Section 4.1. However, the implementation may bring topological collapses in generating triangulation on some data sets (see buildings encircled in Figure 4.6) due to some unknown exceptions in handling geometries by the Java implementation of the algorithm where the cause could not be found. Thus, this polygon triangulation approach is not used in this research to derive explicit neighbourhood relations between polygons.

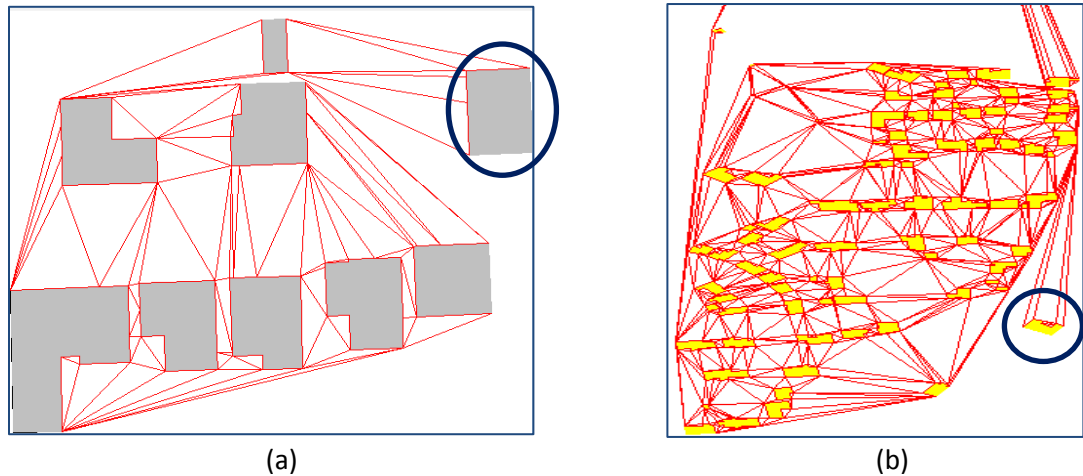


Figure 4.6 (a) A simple data set of buildings and (b) a larger data set with buildings irregularly spaced, triangulated based on the polygon triangulation.

4.4 Testing of a new constrained algorithm on Delaunay triangulation

The constrained algorithm implemented in this research on the polygon triangulation provides rich neighbourhood relations with existing building vertices when comparing the

results with those obtained using the CDT with Poly2Tri library given in Table 4.1. However, an improved algorithm is required since the implementation creates exceptions in triangulation as mentioned in the previous Section 4.3.3. For this purpose, the Delaunay triangulation with so-called recursive edge-flipping technique (Berg *et al.*, 2008) is used to satisfy Delaunay's condition applied to triangles formed from a set of points. It is implemented based on the incremental method as explained in Section 2.4.1. The approach used to constrain edges of polygons is based on the edge deletion technique discussed by Shewchuk (1999) in Section 2.4.2. Initially buildings are enriched based on the contextual features such as road and river network. A field called `global_id` is created and assigned to each region (partition) to identify buildings surrounded by the contextual features in the prototype development (Appendix B.3). Then each building in a region is assigned a unique local IDN. Then testing of algorithms is performed on building polygons falling within each region (see region surrounded by the roads delineated in Figure 4.7). In addition to reading building geometries from the ASCII format, the graphical user interface (GUI) of the prototype has the ability to read PostGIS geometries directly from the PostgreSQL database system (Appendix B.3).

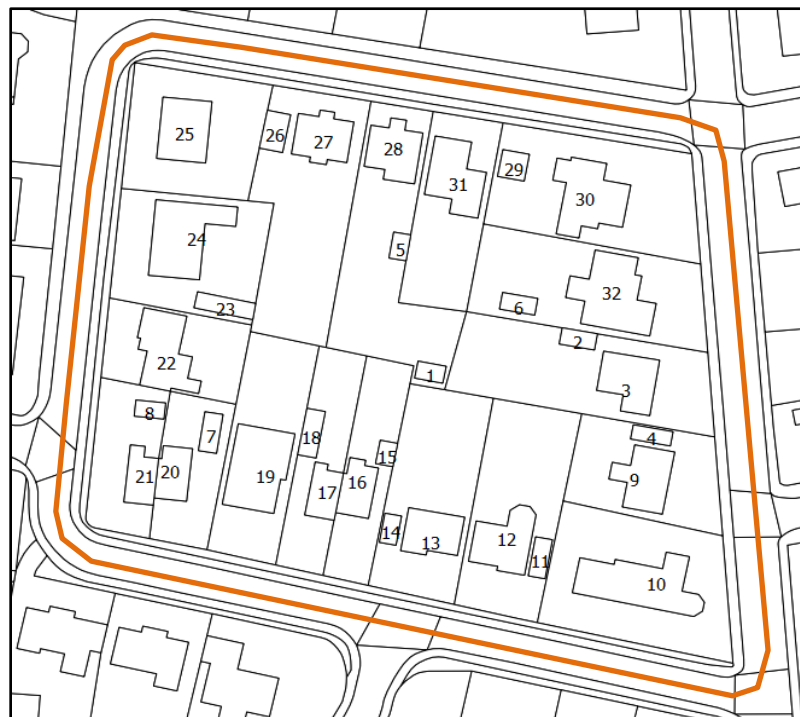


Figure 4.7 Building polygons with local IDNs in a single region surrounded by the road network.

4.4.1 Input data structure

The input data structure is the same as the one used in the CNDT and the polygon triangulation described in Sections 4.2.1 and 4.3.1. The same prototype, as given in Appendix A.2, is used for input data file creation.

4.4.2 Triangulation algorithm

This algorithm falls into the category of an incremental insertion algorithm as described in Section 2.4.1 (see Appendix G.1 for the pseudo code). The triangulation initially starts with a single triangle. Then the points are incrementally inserted one by one into the interior or the exterior to the initial triangulation. The duplicate points are ignored when adding new points to the triangulation. The sequence of steps of the constrained algorithm on Delaunay triangulation is as follows:

- Extract vertices of all building polygons as input points in the triangulation known as the site points P with vertex IDN of each point as building polygon IDN (local IDN).
- Start the initial triangulation with a single triangle with its three points. This triangle is called the “starting triangle”.
- When inserting a new point p also known as Query point p to the triangulation initialised with the “starting triangle”, the point location is determined from the convex hull of the existing site points P , so as to see if the new point is inside or outside the existing triangulation Δ_N .
- If the new point p is inside, then the triangle t of Δ_N containing p is located by starting the search from the “starting triangle” and then looking for the next triangle t according to the relation between the query point p and the triangle edges. Then split triangle t into three triangles by making three new edges between p and the nodes of t to obtain the new triangulation Δ'_{N+1} (Figure 4.8(a)).

- If the query point p lies outside Δ_N , the initial triangulation Δ'_{N+1} is obtained by connecting p to all nodes at the boundary of Δ_N that are visible from p (Figure 4.8(b)).
- If the query point p falls on an edge of an interior triangle or a half-plane triangle bounded by the convex hull, the old edge is replaced by two edges (Figure 4.8(c)), which is not a split of the old edge in the case of an interior triangle and in the case of a triangle bounded by the convex hull (half-plane triangle), it is split into two by adding a new edge (Figure 4.8(d)) to form the initial triangulation Δ'_{N+1} .

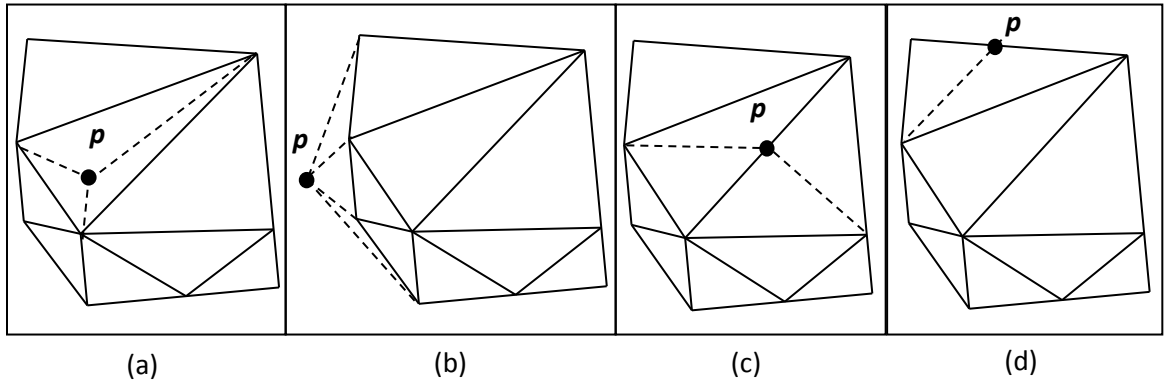


Figure 4.8 Insertion point location: (a) inside triangle (b) outside convex hull of Δ_N , (c) on an edge of an interior triangle of Δ_N and (d) on an edge of a triangle bounded by convex hull to form the initial triangulation Δ'_{N+1} .

- Once the initial triangulation Δ'_{N+1} is formed, the next step is to apply the swapping procedure based on circumcircle test to swap edges in Δ'_{N+1} until all edges are locally optimal, and the default triangulation Δ_{N+1} is Delaunay. The circumcircle test checks if a triangle t in triangulation Δ_N requires modification when inserting a new point p to obtain the Delaunay triangulation Δ_{N+1} if and only if the circumcircle of t contains point p in its interior.
- The swapping procedure runs recursively starting from the three initial edges of the “starting triangle” incident with point p without examining all edges of the triangulation Δ_N (Figure 4.9).

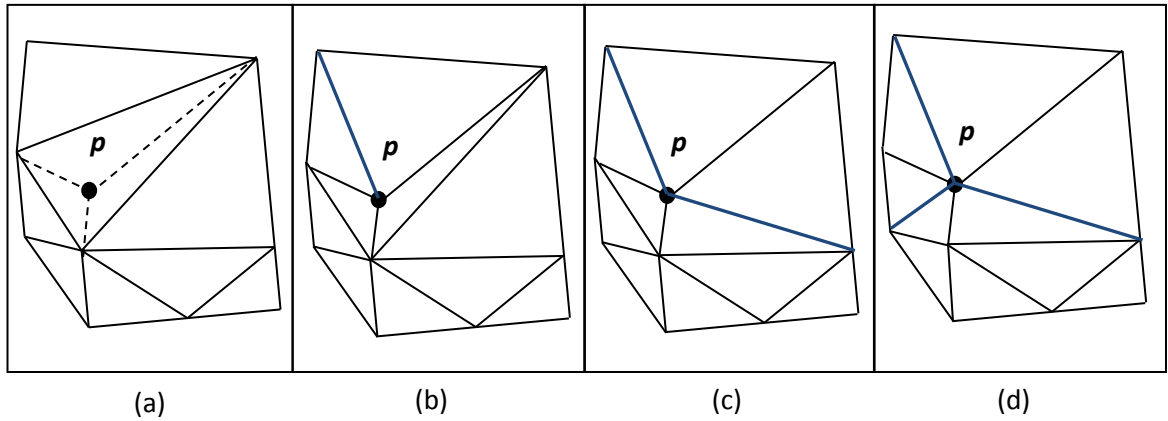


Figure 4.9 Swapping procedure when inserting a point p into Delaunay triangulation in (a) each picture shows the triangulation after a new edge has been swapped from (b) to final Delaunay triangulation Δ_{N+1} in (d).

- Once the default Delaunay triangulation is complete on the set of site points which are the building vertices, the next step is to identify and delete triangle edges that run across polygon edges used as constraint edges (Shewchuk, 1999) and building triangle edges (Figure 4.10(a)) from the default triangulation array since no constraints were applied during default Delaunay triangulation.
- In order to identify the edges of such triangles, 2D topological relations are used. Both intersection and cover (Egenhofer, Litwin and Schek, 1989) relations are checked between each triangle in the triangulation array against all building polygons in the data set. This is a time-consuming process when the data set is very large and, therefore, spatial indexing is used to improve the efficiency. The spatial indexing used is based on the STR algorithm (Leutenegger, Lopez and Edgington, 1997) which is a query-only R-Tree with packing for 2D data implemented in JTS.
- However, one of the disadvantages in R-Tree is the poor performance in unduly retrieving a large number of nodes in order to satisfy a query. The packing algorithm -STR - pre-processes the data to be stored in an R-Tree so that fewer nodes are accessed while performing a query thus improving space utilisation and query time. Therefore, from the STR, buildings which intersect a given query region which is a triangle in the default triangulation array are retrieved very fast for

checking topological intersection and cover with the queried region. Such retrieved buildings are a small subset of all available buildings in the data set.

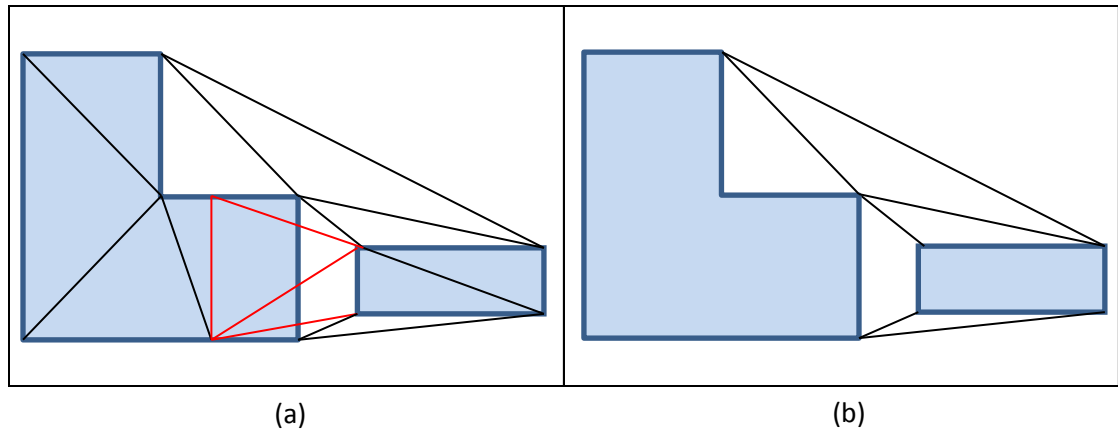


Figure 4.10 Default Delaunay triangulation: (a) triangulation with crossing triangles (red) over building polygons and (b) triangles left after removing crossing triangles and building triangles (triangles inside building polygons in Figure 4.11) that intersect and cover building polygons respectively.

- If such a crossing triangle or a building triangle (Figure 4.10(a)) is found, the triangle is removed from the triangulation array. This topological intersection/cover search criterion is checked for all triangles in the triangulation array finally to remove all crossing and building triangle edges.

Figure 4.11 illustrates three types of triangles that can occur in a Delaunay triangulation according to Haowen, Weibel and Bisheng (2008). A “building triangle” connects the vertices of the same polygon and lies inside the polygon, a “false triangle” connects the vertices of the same polygon but lies outside the polygon and a “true connection triangle” connects two or more polygons in the space region which is the area between the convex hull of all the polygons and the union of polygons.

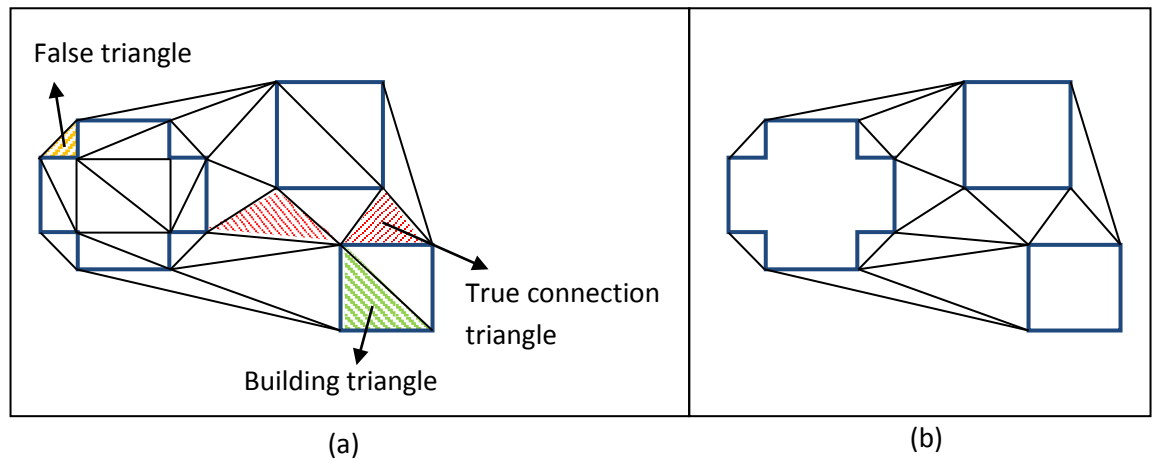


Figure 4.11 Representation of Delaunay triangulation: (a) default Delaunay triangulation considering vertices of all buildings as the site points and (b) the same triangulation after removing all building triangles, based on Haowen, Weibel and Bisheng (2008).

The removal of both crossing triangles and building triangles from the final triangulation array is required to obtain valid proximity relations between building geometries. Removal of crossing triangles leaves the triangulation invalid in the sense that every point falling in the space region R which is the area obtained by subtracting the summation of area of each building polygon from the convex hull of all site points P , does not belong either to an edge or a vertex of one or more triangles or to the interior of a single triangle. This is clear from the trapezoidal area created after the removal of crossing triangles from the triangulation in Figure 4.10 (b).

- Next step is to identify the polygons which need to be re-triangulated due to the removal of crossing triangles created in the initial default Delaunay triangulation steps. The re-triangulation process is explained in Figure 4.12. Triangles generated caused by the removal of crossing triangles after subtraction (see hatched triangle (middle) in Figure 4.13) are not re-triangulated and added to the triangle array to include missing triangles of the region R . The other convex and/or concave polygons (Figure 4.13) with four sides or more are re-triangulated and added to the triangulation array to complete the triangulation process. In this polygon re-triangulation process, crossing triangles formed as a result of concave polygons are required to be searched again and removed if found. Once the triangulation process is complete, the union of all triangles in the constrained triangulation must

be equal to the region R over which the triangulation is defined according to the rules of validation of a triangulation explained in Section 2.4, that is, $R = \bigcup t_{i,j,k}$ where i, j and k refers to vertex IDNs of the triangle t .

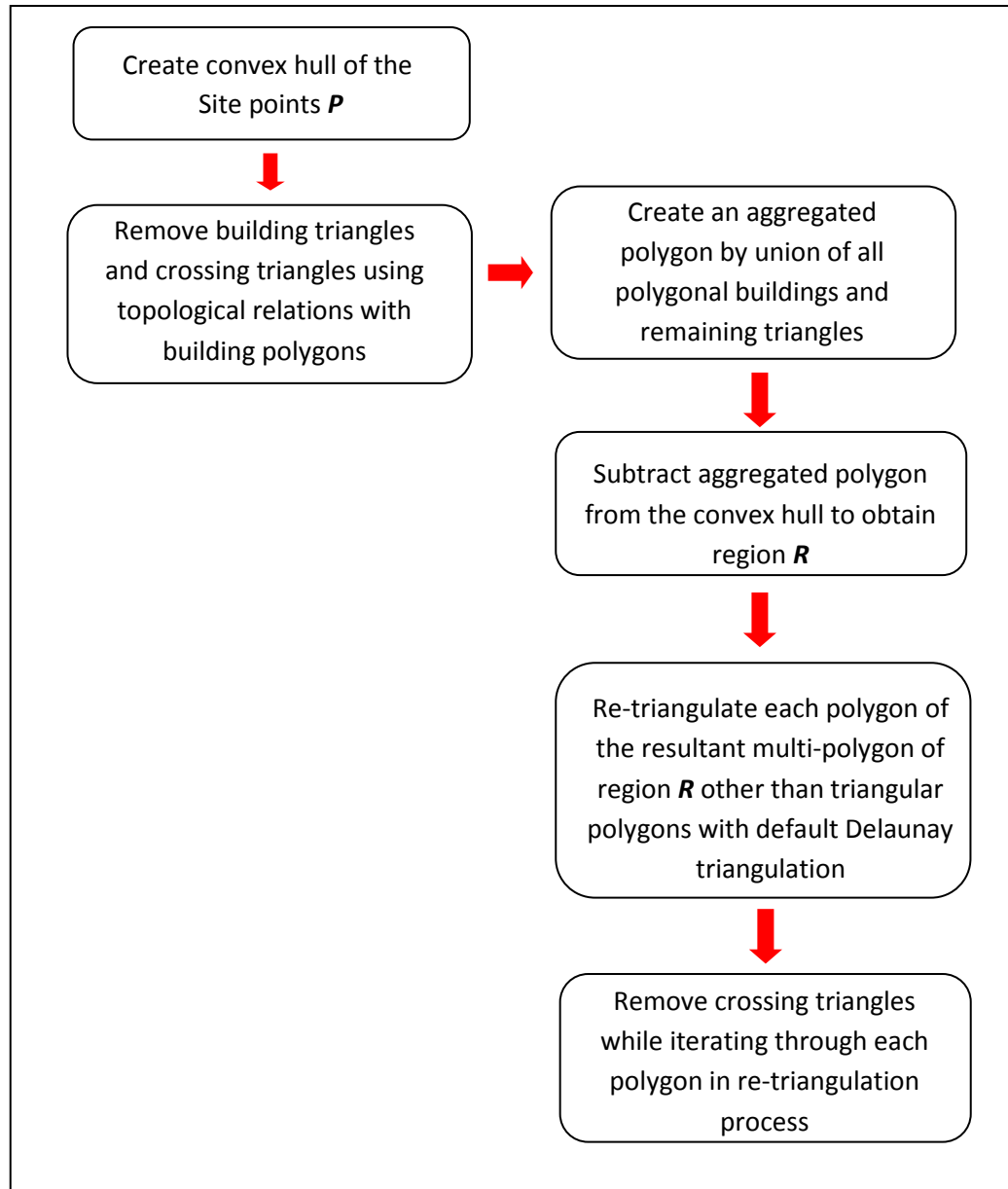


Figure 4.12 Re-triangulation steps of the isolated polygons after subtraction of triangles and building polygons from the convex hull of all the site points P .

- It is important to note that the triangulation handles typical cases of site point configuration in the following manner: (a) no triangle is created if three points are co-linear (b) duplicate points are ignored in the triangulation and (c) if the four points on a convex quadrilateral are co-circular, the choice of the triangulation depends on the point order of each triangle.

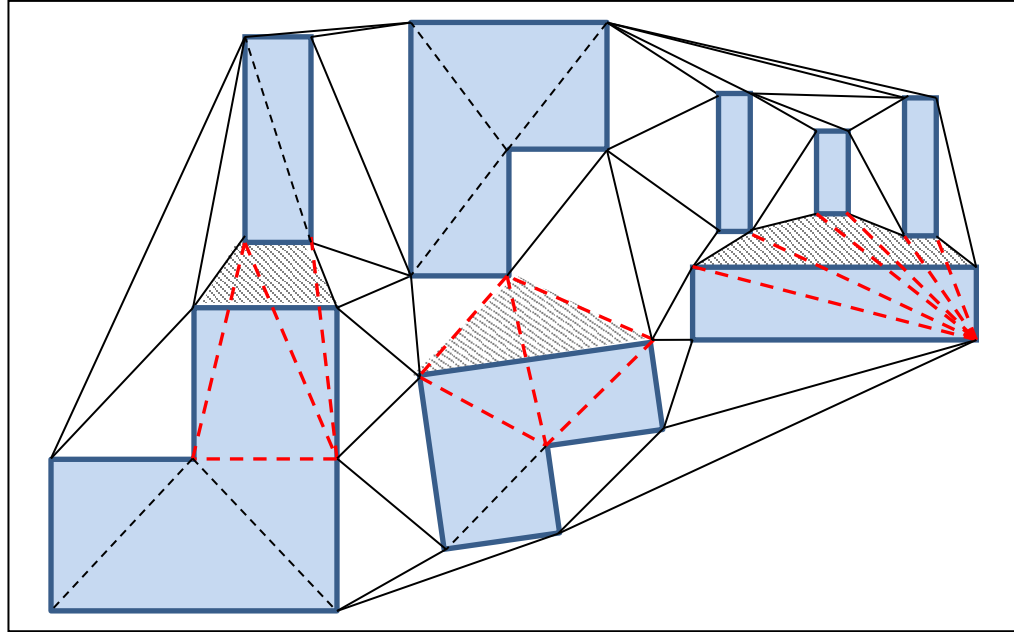
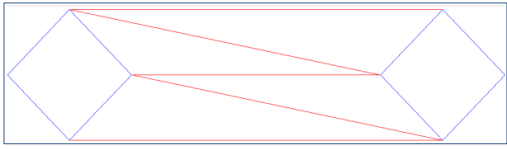
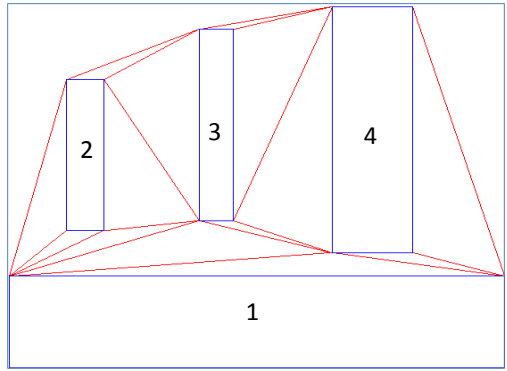
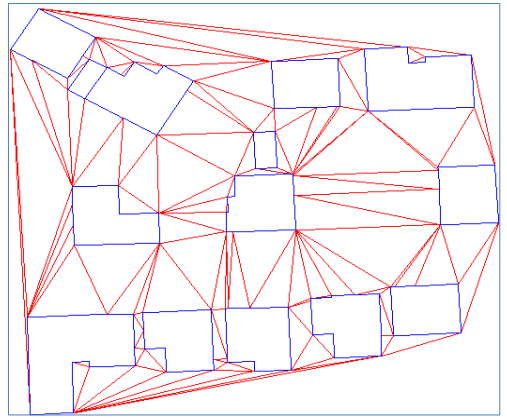
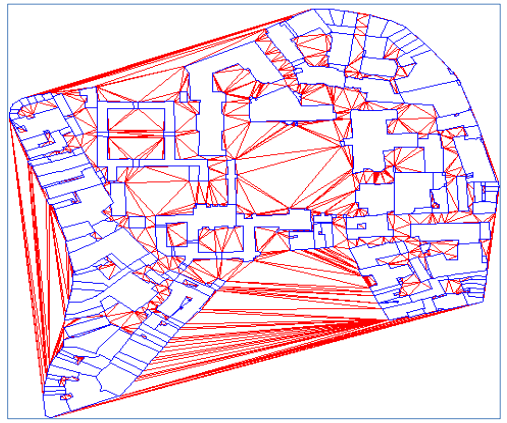


Figure 4.13: Crossing triangles to be removed in the triangulation process are shown in red broken lines. Remaining isolated polygons hatched including a convex polygon (left), a triangle (middle) and a concave polygon (right) after the removal of initial crossing triangles

The results of the implemented constrained triangulation algorithm on the data sets used for testing are given in Table 4.4.

Table 4.4 Results of the constrained triangulation developed in this research based on the Delaunay triangulation with the recursive edge-flipping technique (Berg *et al.*, 2008) using the incremental method.

#	Type of data	Results of the constrained triangulation
1	Synthetic data set represented in Figure 4.1(a)	
2	Synthetic data set comprising of narrow buildings with long and parallel edges	
3	OS MasterMap source data (1 : 1.25K) comprising of edges of all buildings with intermediate vertices except buildings 2, 12 and 14	
4	OS MasterMap Data (1 : 1.25K) - London-part of	

4.4.3 Proximity links derivation between polygons

In the constrained triangulation approach based on edge deletion in Section 4.4.2, the minimum distance between each linked pair of buildings is obtained from the triangles generated between buildings (true connection triangles connected between buildings only) by the use of the data structure adopted in the triangulation with each building IDN represented as site point IDN based on the nearest neighbour links formed by the triangulation. The steps in deriving proximity links are as follows:

- Iterate through each triangle in the final constrained triangulation array and retrieve the building IDNs linked to each node of the triangle with the use of the Hashtable data structure illustrated in Figure 4.1 in Section 4.1.
- Find pairs of building link relations between each edge of the true connection triangles (Figure 4.11 (a)) representing building neighbourhood relations with the help of building link IDN information at three corner points of the respective true connection triangle (Figure 4.14 (b)). This process also takes into account the buildings that are vertex-contiguous and/or edge-contiguous in addition to detached buildings. In this process, if two nodes of a triangle have the same building IDN, such links are ignored. Similarly, 'False triangles' (Figure 4.11 (a)) representing links between vertices of the same building are ignored while iterating the triangulation array.
- From each pair of building link relations, compute the minimum Euclidean distance between each pair of buildings and append the result with such pairs of building links of each true connection triangle into the one-dimensional array representing the proximity.
- Now the array has all possible links of neighbourhood relations of all buildings, including duplicate links (Figure 4.14 (c)).

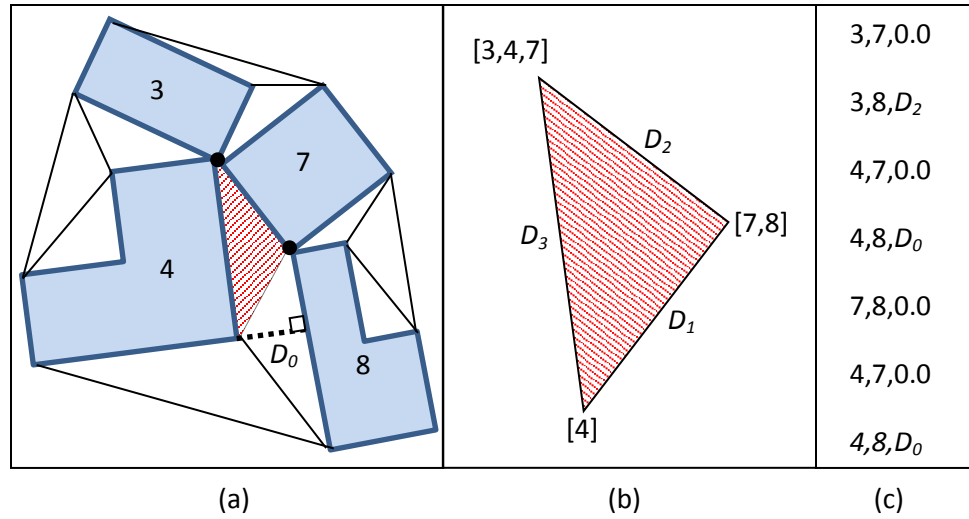


Figure 4.14 (a) Constrained triangulation with duplicate nodes of a triangle hatched in red colour (b) same triangle with duplicate node IDNs and distances of the three edges D_1 , D_2 and D_3 and (c) array representing proximity links of both contiguous and disjoint buildings with the minimum Euclidean distance.

4.4.4 Evaluation of the results of the constrained algorithm on Delaunay triangulation

When considering the results #1, #2 and #3 in Table 4.4, the triangulation represents explicit neighbourhood relations between the buildings. Further, the triangulation does not split edges inserting new vertices to maintain the Delaunay property (compare the result #2 between building 1, and 2, 3 and 4, and the result #3 between building IDNs 5 and 6, 7 and 8, 8 and 9, and 11 and 12 in Table 4.4 with the results #2 and #3 generated using the CNDT of the same building data sets in Table 4.2). It can also handle very complex buildings that share a common edge (building IDNs 2 and 3 in result #3 of Table 4.4) and contain holes (result #4 in Table 4.4). It is also observed that the two new constrained triangulation algorithms - algorithm based on polygon triangulation as described in Section 4.3 and the algorithm based on Delaunay triangulation with edge deletion technique as described in Section 4.4 - provide the same results for the three data sets used to test all four algorithms. However, the algorithm based on the edge deletion technique is more robust compared to the algorithm developed based on polygon triangulation in terms of handling neighbourhood relations (topological relations).

Since the algorithm represents explicit neighbourhood relations, it can be used to derive the Gestalt factors such as proximity, orientation and shape between building polygons under data enrichment process for subsequent generalization.

The tests to generate a constrained triangulation based on the Delaunay triangulation with edge deletion technique and its neighbourhood relations with proximity have been performed on a Laptop with an Intel Core 2 Duo 2.5 GHz processor and a 4GB of RAM (3.5GB usable). The default Delaunay triangulation spends 110ms while constrained triangulation spends 1200ms on a data set consisting of 125 building features. Table 4.5 shows the execution times required to build triangulation and neighbourhood relations over a large set of features.

Table 4.5 Computation times to generate triangulation and neighbourhood relations.

Geometric entities	Triangulation		Neighbourhood relations	
Features/Nodes	Default Delaunay triangulation (ms)	constrained triangulation (ms)	Number of Proximity links	Proximity links (ms)
125/1015	110	1200	364	560
250/2156	155	2370	751	1750
500/3522	170	3460	1504	3800
1000/5879	245	5200	3052	9140

4.4.5 Validation of the constrained algorithm on Delaunay triangulation

Table 4.6 presents the results of the validation of the constrained triangulation algorithm developed in this research for subsequent data enrichment and generalization processes. It is validated with a synthetic data set which provides exceptions in topology in the constrained triangulation as discussed by Ware and Jones (1996) and Ai *et al.* (2007), and with OS MasterMap data sets at the scale of 1 : 1.25K.

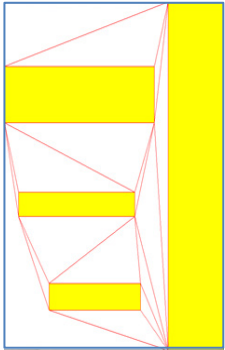
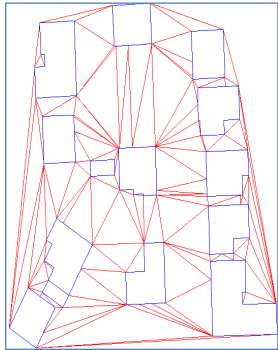
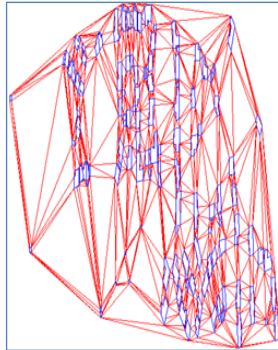
#	Constrained triangulation output	Area of the convex hull (Ach) in m ²	No. of Buildings	Area of the union of Buildings (AuB) in m ²	Domain of triangulation (Ω) in m ² $\Omega = \text{Ach} - \text{AuB}$	No. of triangles	Area of the union of Triangles (AuT) in m ²
1		8544.50	4	4442.00	4102.50	14	4102.50
2		3840.83	14	1208.17	2632.66	121	2632.66
3		90052.85	125	9211.72	80841.13	1170	80841.13

Table 4.6 Validation of the constrained triangulation developed using the edge deletion method (Shewchuk, 1999) with #1) a synthetic data set comprising of narrow and long buildings with parallel edges, and OS MasterMap data sets #2 and #3 shown above.

The validation is based on the criteria defined to check the triangulation validity as explained in Section 2.4. The domain (Ω) of the constrained triangulation is the space region which is the free space between buildings and the convex hull of the entire building polygon vertices. With the pictorial representation and calculation of the area of the domain and comparison of the total area of the generated triangles in the domain, all the criteria given in the aforesaid Section 2.4 are satisfied, making constrained triangulation a valid triangulation.

4.5 Outcome of the triangulation algorithms used for testing

The intention of this section is to explain the outcome of each of the four triangulation methods tested for generating the constrained triangulation and clarify the terminology of new algorithms developed.

4.5.1 Constrained Delaunay triangulation

The CDT based on the sweep line algorithm by Domiter and Žalik (2008), implemented in the open source poly2Tri triangulation library as described in Section 4.1, can only process polygon geometries in generating triangulation. The triangulation does not handle polygons that share common boundaries either. Further, this triangulation is not Delaunay stable since it does not respect the empty circle criterion adopted in the Delaunay triangulation as described in Section 2.4.2. As a consequence, it creates skinny triangles thereby generating implicit topological relations between polygons as evident from the visual result #3 in Table 4.1 (building IDN 11 is not connected to building IDNs 6, 7 and 8).

4.5.2 Conforming Delaunay triangulation

The CNDT which is implemented using the JTS algorithm library based on the incremental algorithm by Ruppert (1995) as described in Section 4.2 can process both line and polygon geometries in generating triangulation with Delaunay stable triangles with more hooks between geometries with the addition of Steiner points. Thus, it can create explicit neighbourhood relations between features. However, these points are not part of the

geometry of the input source data set. This implementation also has the ability to enforce edge constraints in the triangulation (see results in Table 4.2).

4.5.3 New constrained algorithm on polygon triangulation

The new constrained triangulation algorithm developed in this research uses the open source Java source code of the polygon triangulation algorithm based on classic ear-clipping by Eberly (2008) as discussed in Section 2.4.4. The algorithm can only be used with polygon geometries. Further, the polygon edges can be constrained in generating the triangulation. It can also handle buildings that share a common edge (buildings with IDNs 2 and 3 in the result #3 of Table 4.3). Also, the triangulation generates triangles that connect only the vertices of the input data set.

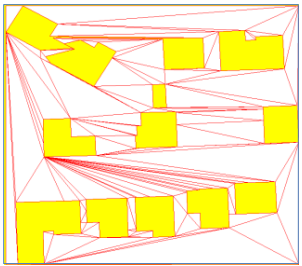
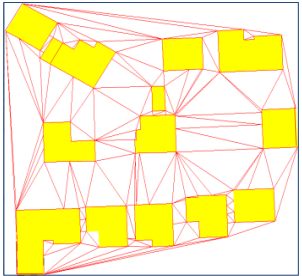
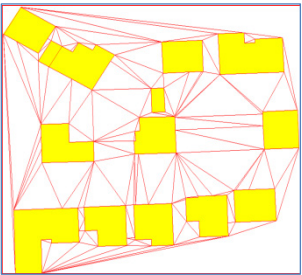
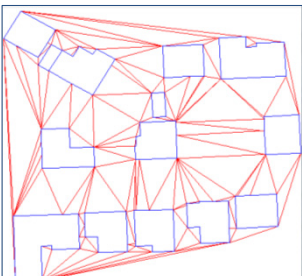
4.5.4 New constrained algorithm on Delaunay triangulation

This algorithm employs Delaunay triangulation with so-called recursive edge-flipping technique (Berg *et al.*, 2008) to satisfy Delaunay's condition applied to triangles formed from a set of points. It is implemented in this research based on the incremental method as explained in Section 2.4.1. The algorithm can only be used with polygon geometries. The approach used in this algorithm to constrain edges of polygons is based on the edge deletion technique discussed by Shewchuk (1999) as described in Section 2.4.2. It can also handle buildings that share a common edge (building IDNs 2 and 3 in result #3 of Table 4.4). Further, the triangulation generates triangles that connect only the vertices of the input data set. Since this algorithm has the ability to generate Delaunay stable triangles with applying edge constraints, it is termed as the Delaunay constrained triangulation (DCT) in this research.

4.5.5 Comparison of the triangulation algorithms

Table 4.7 presents a comparison of the four algorithms on the constrained triangulation structures - two existing algorithms and two new modified algorithms - with their properties, capabilities and runtime exceptions in this research.

Table 4.7 Different triangulation types implemented and compared. Algorithms are: (1) Sweep line algorithm by Domiter and Žalik (2008) (2) incremental algorithm by Ruppert (1995) (3) new algorithm (see Section 4.3) coupled with polygon triangulation algorithm by Eberly (2008) and (4) new algorithm (see Section 4.4) coupled with incremental point insertion based on the recursive edge-flipping technique by Berg *et al.* (2008).

#	Triangulation data structure	Visual representation	Neighbourhood relations	Handling shared geometry	Addition of new vertices	Delaunay property	Runtime exceptions
1	CDT*		Implicit	No	No	Not preserved and skinny triangles generated	Cannot handle shared edges
2	CNDT*		Explicit	Yes	Yes. Steiner points added	Preserved	No
3	Constrained triangulation* on polygon triangulation approach		Explicit	Yes	No	Preserved	Collapses in topology
4	DCT**		Explicit	Yes	No	Preserved	No

* existing algorithm

** Modification and/ or extension to existing algorithm

When comparing the results of the constrained algorithm based on polygon triangulation with the constrained algorithm based on the Delaunay triangulation (termed as DCT), both algorithms have produced the same results (compare result #3 in Tables 4.3 and 4.4). Thus, it is evident that the constrained algorithm based on the polygon triangulation is also Delaunay stable. Out of the four algorithms, there are two potential algorithms - algorithm #2 (CNDT) and algorithm #4 (DCT) in Table 4.7 - that can be used to derive spatial relations between polygon geometries, between line geometries and between polygon and line geometries, and between polygon geometries respectively in any spatial application. The algorithm #2 adds Steiner points that are not available in the source data used to generate triangulation, allowing more hooks between feature geometries.

4.6 Conclusion

The chapter presents and describes how the existing triangulation algorithms have been tested, analysed and modified with the synthetic and the real data sets in order to identify and/or develop constrained triangulation structures in this research to derive explicit neighbourhood links between polygons, between lines and between polygon and line objects for the data enrichment process and the subsequent automatic map generalization. The research has identified that the CNDT implementation in JTS based on the work by Ruppert (1995) can derive the explicit adjacency relationships between polygons, between the lines, and between lines and polygons with enforcing edge constraints. In deriving this triangulation, Steiner points are introduced that are not part of the source data set. Therefore, an algorithm has been developed in this research for the enrichment of Steiner points (Section 4.2.2) in order to track the polygon or line geometry to which each Steiner point created belonged. The new constrained algorithm for polygon triangulation based on classic ear-clipping by Eberly (2008), as discussed in Section 2.4.4, which is developed in this research with the JTS library, gives explicit adjacency relationships between polygons while enforcing edge constraints using input points in the source data set. However, the implementation introduces topological collapses in generating triangulation on some data sets (see Figure 4.6). Further, the new modified constrained algorithm based on Delaunay triangulation (addressed hereafter as DCT as termed in Section 4.5.4) developed with the existing algorithms - edge flipping technique by Berg *et al.* (2008) and edge deletion technique discussed by Shewchuk (1999) - can derive explicit adjacency relationships between polygons while enforcing their edges as constraints using the input points in the source data set without introducing Steiner points. Next chapter will describe in detail the implementation of the tools for the building data enrichment process by the use of the developed DCT structure.

Chapter 5 Implementation - II: Spatial Data Enrichment Process

This chapter presents the data enrichment processes required for automatic map generalization. The two main phases of data enrichment in this work are: (a) spatial clustering and (b) shape enrichment of spatial clusters. It further describes how existing algorithms on these two phases - creating spatial polygon clustering and enhancing shape characteristics of such derived clusters based on building geometries - are tested and evaluated by modifying existing algorithms to develop new algorithms required for the automatic map generalization process. For testing and evaluation (internal validation), both synthetic and real data sets will be used. The testing platform is open source Java object oriented programming language with data stored in PostGIS, a spatial extension to PostgreSQL object-relational database management system to handle spatial data.

5.1 Hierarchical polygon clustering process

The polygon clustering algorithm developed in this research is based on the hierarchical clustering approach by Qi and Li (2008), which is similar to the manual clustering of cartographers as mentioned in Section 2.3.3. In the manual clustering approach, the global constraints can be organised hierarchically in a spatial database or in a topographic map. For example, roads between cities can be classified as national highways, and those within a city can be classified as major roads, minor roads and cul-de-sacs. Similarly, a hydrographic network can be identified as main rivers and tributaries at different levels. Figure 5.1(a) illustrates a hierarchical structure of a road network. In the manual building grouping process, roads and hydrographic features with corresponding levels of details are chosen for the initial partition of buildings into regions according to the target map scale. Further, the use of Gestalt factors considered as local constraints - proximity, common orientation and similarity - is also hierarchical in the manual grouping of buildings (see Figure 2.14 in Section 2.3.3). This can be clearly described with the help of Figure 5.1(b). Five building groups (**i**, **ii**, **iii**, **iv** & **v**) can be identified initially according to the degree of proximity between buildings. Groups (**i**) and (**v**) will not be clustered anymore because the degree of proximity between buildings in these two groups is very

close. Groups (iii) and (iv) can be considered as separate single building clusters since both are very far from each other and from other groups. Building group (ii) delineated in red broken lines can be considered as a single group since the degree of proximity between the buildings is medium. The buildings in this medium distance range group (ii) are further clustered based on the differences in orientation between buildings into ‘small’ and ‘large’ categories. Since the differences in orientation between buildings in the group (ii) are small, these are categorised into a single sub-group. Finally, this single sub-group is further clustered, considering the differences in similarity between buildings in shape labelled as ‘similar’ and ‘dissimilar’ to end up with final building groups **x**, **y** (**y** is a single building) and **z** (**x** and **z** delineated in black broken lines). All these three building groups (x, y and z) are dissimilar in shape and have an almost similar orientation with the proximity between two buildings being within the medium distance range. Figure 5.2 illustrates the hierarchical relationship of these three local constraints.

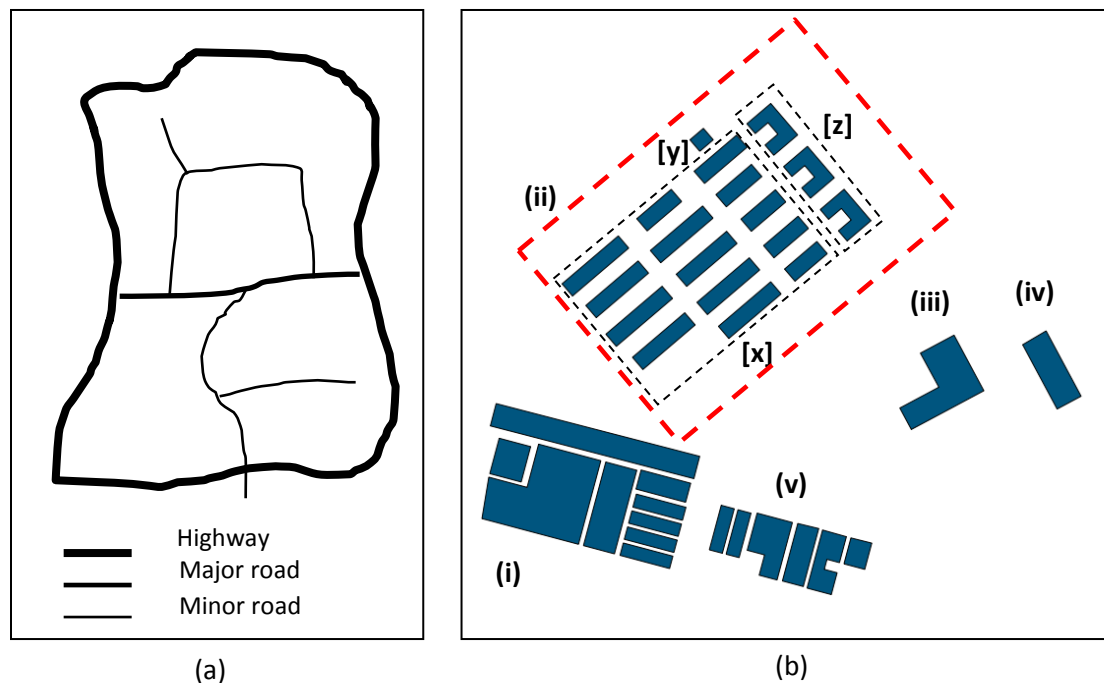


Figure 5.1 (a) Hierarchical structure of the road network (part) and (b) manual process of building grouping, based on Qi and Li (2008).

The following subsections will test and compare the manual process of this hierarchical clustering approach with the same process automated using different algorithms in deriving Gestalt factors - proximity, orientation difference, and similarity difference in shape - contributing to creating clusters coupled with a graph-based minimum spanning tree (MST) clustering algorithm.

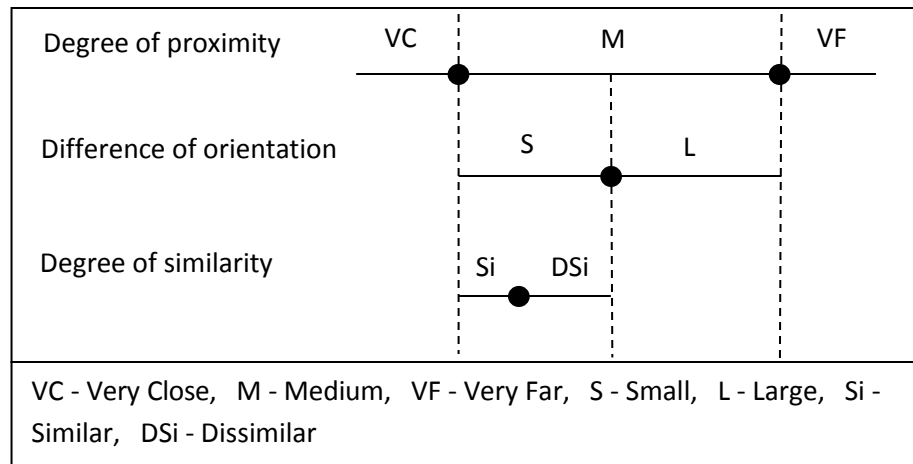


Figure 5.2 Hierarchical relationship between the three local constraints for building grouping, based on Qi and Li (2008).

5.2 Automation of hierarchical clustering process

Clustering of buildings is applied to each partitioned region created with the use of global constraints (contextual features) as described in Section 5.1 in the data set. In the process of building clustering within a region, hierarchical local constraints - proximity, common orientation and similarity in shape - are used as weights in the minimum spanning tree (MST), as discussed in Section 2.3.3. This is implemented using Prim's algorithm (Prim, 1957) (see Appendix C.1 for the pseudo code of the algorithm). This clustering process with full automation is similar to the manual process of building clustering by cartographers. The building clustering process can principally be divided into three hierarchical sections: (a) clustering based on proximity (b) clustering based on orientation difference and (c) clustering based on the similarity difference in each partitioned region.

5.2.1 Derivation of the Gestalt factors

Proximity

The proximity relation between each pair of buildings is generated by the Delaunay constrained triangulation (DCT) algorithm developed in this research (see Section 4.4).

Orientation difference

Duchêne *et al.* (2003) have discussed five measures for building orientation - longest edge, weighted bisector, wall average, statistical weighting and MBR for building orientation. There are basically two types of orientations defined for a building: (a) general orientation used to characterise the elongation of a building and (b) the orientation of the walls of a building. Of the five measures, the MBR and the weighted bisector represent general orientation, and all other three represent wall orientation (Figure 5.3).

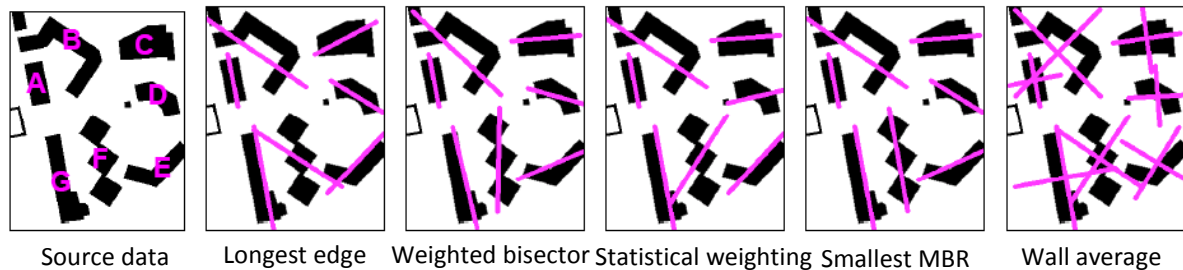


Figure 5.3 Existing measures of building orientation, from (Duchêne *et al.* (2003).

Duchêne *et al.* (2003) have concluded that the MBR is the most appropriate to define the general orientation of a building by testing the measures on a large set of buildings data. Usually, the MBR is computed to have an idea of the extension of the object (Toussaint, 1983). The orientation of the longest side of MBR is assigned as the general orientation of a building. However, if the MBR is used to calculate the general orientation, the orientation of a square building (Figure 5.4(b)) and a circular building (Figure 5.4(c)) cannot be defined. And the orientation of an offset terraced building (Figure 5.4(e)) is quite extensive along the longest side of the MBR (red broken line) where the orientation of the walls would be more appropriate (orange thick lines).

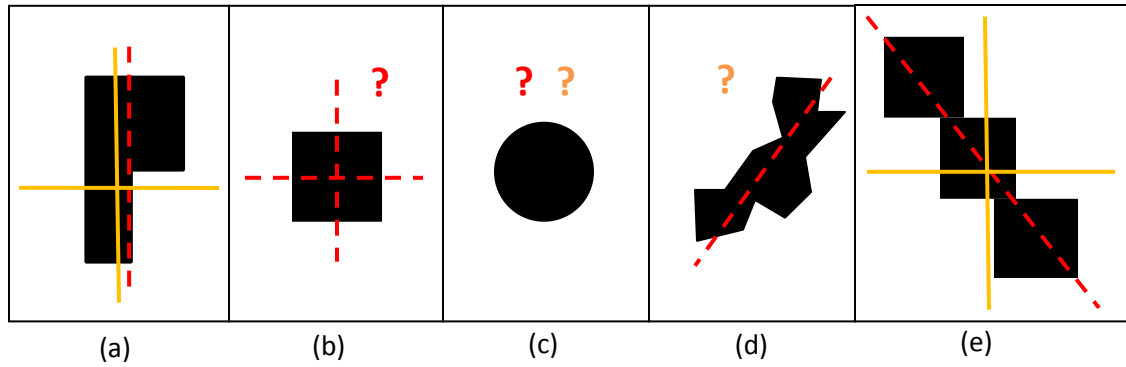


Figure 5.4 Smallest minimum bounding rectangle (SMBR): (a) SMBR = Walls (b) SMBR non-consistent (c) SMBR and Walls non-consistent (d) Walls non-consistent and (e) SMBR \neq Walls, based on Duchêne *et al.* (2003).

Li *et al.* (2004) have used the MBR algorithm with the contribution of both major and minor axes to define building orientation for subsequent building clustering. This approach can deal with computing orientation of buildings with square shapes where the directional extent of a building is not perceptually clear (Figure 5.4 (b)). The use of both axes of buildings has also been adopted for building clustering by Haowen, Weibel and Bisheng (2008). However, circular buildings and terraced buildings still cannot be handled using this approach. Furthermore, the longest side of the MBR as experimented by Duchêne *et al.* (2003) has been used by Qi and Li (2008) for building clustering despite this approach not being suitable for the building shapes shown in Figure 5.4(b), (c) and (e).

When reviewing the literature, the MBR with the contribution of the major axis and the minor axis to define a common orientation approach by Li *et al.* (2004) and a wall orientation method by Duchêne *et al.* (2003) are both suitable for polygon clustering. However, if a wall statistical orientation algorithm (see Appendix C.2) is used for building polygon clustering, it would only have the exception for buildings with circular shape since data sets with irregularly shaped buildings (Figure 5.4(d)) are more rare. If the MBR with the significance of the major axis and the minor axis is used, it would often end up with two exceptions for circular or terraced shaped building polygons (Figure 5.4(c) and (e)).

Table 5.1 Typical examples of the orientation difference between a rectangular pair of buildings based on the wall orientation algorithm by Duchêne *et al.* (2003).

Case	Orientation calculation with wall orientation (α) of each building	Illustration
I	α_1 and $\alpha_2 > 45^\circ$ Orientation difference = $(\alpha_2 - \alpha_1) < 45^\circ$	
II	α_1 and $\alpha_2 = 45^\circ$ Orientation difference = $(\alpha_2 - \alpha_1) = 0^\circ$	
III	<p>Situation I: $\alpha_1 < 45^\circ$ and $\alpha_2 > 45^\circ$ and $(\alpha_2 - \alpha_1) < 45^\circ$ Orientation difference = $(\alpha_2 - \alpha_1)$</p> <p>Situation II: $\alpha_1 < 45^\circ$ and $\alpha_2 > 45^\circ$ and $(\alpha_2 - \alpha_1) > 45^\circ$ Orientation difference = $(90^\circ + \alpha_1 - \alpha_2)$</p>	
IV	α_1 and $\alpha_2 < 45^\circ$ Orientation difference = $(\alpha_2 - \alpha_1) < 45^\circ$	

Since the algorithm developed by Duchêne *et al.* (2003) calculates the orientation of a building based on its wall statistics with a value forced to be between 0 and $\pi/2$ along with a precision angle increment value (0.25° degrees used in this research), it is an appropriate candidate to compare the orientation difference between a pair of buildings because it can give a proper judgement of the orientation difference based on the visual impression. This is more emphasised when observing the pair of buildings illustrated in Case II, Table 5.1 where the orientation difference between the pair of buildings is 0° when it is calculated by using the wall orientation algorithm. However, when it is calculated based on the MBR, the orientation difference is larger (90° degrees). Therefore, the algorithm by Duchêne *et al.* (2003) based on wall orientation is used to get the orientation difference between each pair of buildings in the clustering algorithm developed in this research.

In the calculation of orientation difference with the wall orientation algorithm using a reference angle of 45° degrees, the algorithm adopted in this research describes how it calculates the orientation difference for the four cases illustrated in Table 5.1. The maximum orientation difference is forced to be a value between 0° and 45° (see Case III in Table 5.1). Adapting this range would enable human beings to perceive the difference between a pair of buildings with ease.

Similarity difference

Li *et al.* (2004) and Haowen, Weibel and Bisheng (2008) have defined two different measures in terms of size and shape based on the ratio of the area and the number of edges respectively in a pair of buildings to find the similarity difference between the two buildings in it. Further, when the orientation of a pair of buildings has no difference, both buildings must be completely similar in shape, size and orientation (Qi and Li, 2008). Based on this fact, using the ratio of area of intersection and the area of the union of the two building polygons superimposed together, the degree of similarity difference has been calculated by Qi and Li (2008). The ratio is an index (value) between 0 and 1, which is a linear estimation of similarity. However, a more robust method together with an improvement in the calculation of the similarity difference by Qi and Li (2008) is developed in this research where the similarity difference between a pair of buildings is

calculated using three values integrated together - symmetric difference, compactness and the Hausdorff distance.

- **Symmetric difference.** This is the area ratio between the non-overlapping portions of the two buildings and the total area of the two polygons. This is a value between 0 and 1 and it represents a linear estimation of similarity. If the two polygons perfectly coincide, the area ratio is 1. The two geometries are superimposed using a translation based on the two centroids of the building polygons before calculating the ratio. Suppose the two buildings are - A and B - then the Symmetric difference can be defined by the equation (1).

$$SD_{A,B} = \frac{S(A \cap B)}{S(A \cup B)} \quad (1)$$

Where, $S(A \cap B)$ is the area of the intersection of the two building polygons A and B, and $S(A \cup B)$ is the area of the union of building polygons A and B.

- **Compactness.** This is defined as the area to perimeter ratio ($C_{A, B}$) of the two polygons. The compactness is a value between 0 and 1 and it represents a linear estimation of similarity. If the two polygons perfectly coincide, its value becomes 1.
- **Hausdorff distance.** The Hausdorff distance (HD) measures the greatest local deviation of two geometries (Atallah, 1983; Günter, 1991). The two geometries are superimposed using a translation based on the centroids of the building polygons before calculating this distance value. In order to compute this value, the discrete Hausdorff distance (DHD) implementation in JTS is used. This function itself has a method to densify the two geometries if the two geometries are to be discretized. The DHD is an approximation to the HD based on discretization of the input geometry where this distance is the maximum deviation between the discrete points of the two geometries as in the HD calculation (Figure 5.5(a)). The discrete points can be either existing vertices of the geometries (the default) or the geometries with line segments densified by a given value (Figure 5.5(b)). Since no

densification is applied in this work when deriving the similarity difference between an adjacent pair of buildings, the DHD thus calculated becomes the HD.

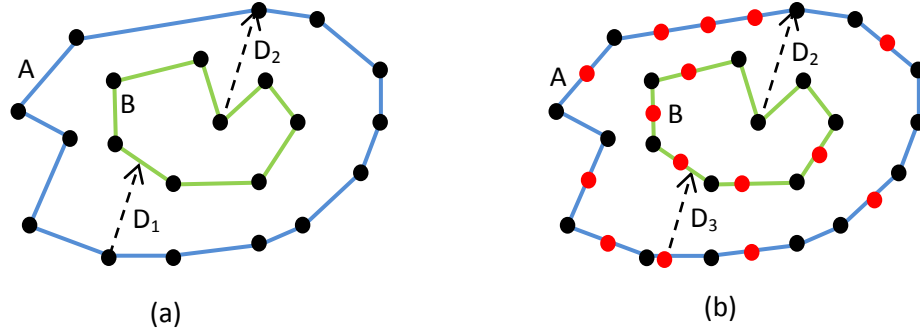


Figure 5.5 Calculation of the Hausdorff distance: (a) distance D_1 is the maximum of the distances from vertices of polygon A to any point in polygon B (the closest point in B might not be a vertex) and distance D_2 is the maximum of the distances from vertices of B to any point in A (the closest point in A might not be a vertex). The Hausdorff distance $d_H(A,B) = \max(D_1, D_2)$ and (b) the discrete Hausdorff distance $d_{HD} = \max(D_3, D_2)$ after discretization of points of the two polygons A and B by densification (densified points are in red).

The $HD = 0$ is mapped to 1 in the HD matching index. Any HD which is greater than or equal to 0.5 mm in map scale is mapped to 0 in the HD matching index. Therefore, the calculated matching index and the final similarity difference index depend on the target scale of the generalized map. Any HD which is greater than zero and less than 0.5 mm at map scale is mapped using equation (2).

$$MHD_{A,B} = 1 - HD \times \frac{1000 \times TS}{0.5} \quad (2)$$

Where, $MHD_{A,B}$ is the matching index of HD and TS is the Target Map Scale.

Finally the similarity difference index ($S_{A,B}$) is defined by the equation (3) by integrating three measures into one.

$$S_{A,B} = \frac{SD_{A,B} + C_{A,B} + MHD_{A,B}}{3} \quad (3)$$

Where, $S_{A,B}$ is the Similarity difference index.

5.2.2 Clustering algorithm

The following are the steps of the algorithm developed in this research for the automatic building clustering process. The Gestalt factors described in the previous Section 5.1 are used in this algorithm (see Appendix G.2 for the pseudo code).

a) Clustering based on the proximity:

- i. Construct the Delaunay constrained triangulation (DCT) network of buildings (Figure 5.6(a)).
- ii. Create an adjacency relationship list of buildings by calculating the proximity using the neighbourhood relations.
- iii. Calculate the differences in orientation and similarity between each connected pair of buildings in the adjacency relationship list and attach values to the list itself (Figure 5.6(b)).
- iv. Weight each linked pair of buildings in the adjacency relationship list with the proximity and create the MST (Figure 5.6(c)).
- v. Then a 2D adjacency matrix is created by traversing through the segmentation of the initial MST created based on three distance thresholds - very close, medium distance and very far - depending on the target map scale on which the buildings are supposed to represent (see Section 5.3 for the specifications used in the testing process). Each element of the matrix stores the information of a pair of buildings in the format of [bid_from (IDN 1), bid_to (IDN 2), proximity (dst), orientation difference (od) and similarity difference index (simd)] into a single array. The null elements in the matrix represent no adjacency link with "0,0" (Figure 5.7 (a)). The number of rows is the number of MST segments, and the number of columns is three (3) based on the hierarchy of proximity - very close, medium and very far.

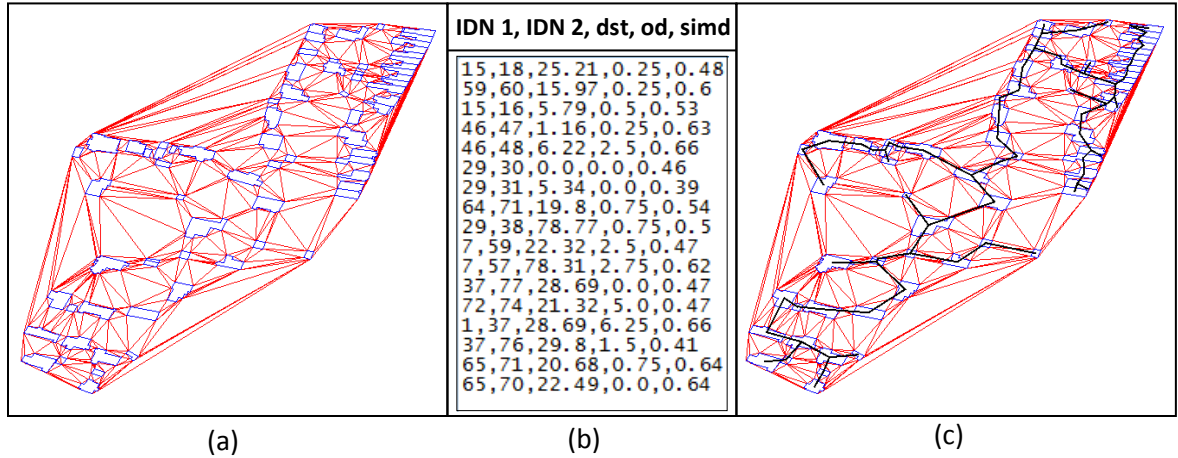


Figure 5.6 (a) DCT (b) adjacency relationship list (part) comprising of [bid_from, bid_to, proximity, orientation difference, similarity difference] and (c) MST segmentation in thick black lines with proximity as the weight where dst: proximity (minimum distance), od: orientation difference and simd: similarity difference index in shape between an adjacent pair of buildings.

vi. The initial adjacency matrix is further refined based on the priority of proximity hierarchy. In the initial adjacency matrix, comparison of common building IDNs between pairs of buildings is carried out in three sequential steps between two columns in the following order - 3 => 1, 2 => 1 and 3 => 2. If the same building IDN is found in two of the columns once compared as in the above order, the building IDN in the higher column is always removed (more priority is given to buildings that are closer).

For example, as depicted in Figure 5.7, links between buildings 1-2, 2-3 and 3-4 belong to the column one ('Very Close') while link 3-5 belongs to column two ('Medium') and link 5-6 belongs to column three ('Very Far') in the initial adjacency matrix according to the distance thresholds. In this matrix, no similar building IDNs are found between the two columns 3 and 1 in the first level of comparison. In the second level of comparison between the two columns 2 and 1, building IDN 3 is available in both columns 1 and 2. Therefore, the building IDN 3 is removed from the second column, leaving building IDN 5 single in the refined matrix [value (3,5,d4,o4,s4) is assigned (0,5)]. In the third level of comparison between the columns 3 and 2, the building IDN 5 is available in both columns. Now considering the priority, the building IDN 5 is removed from the third

column thus leaving building IDN 6 single in the refined matrix (value (5,6, d3, o3, s3) is assigned the value (0,6)). Next, checking is done in each of the columns in the refined matrix to verify such a single building IDN (e.g. IDN 5 stored as (0,5) in column 2 of the refined matrix) is not linked with any other building in the same column. If it is linked, single building IDN 5 is removed from the column element making it null (i.e. 0,0). Once this step is complete, the adjacency matrix is completely refined for the final building clustering based on the proximity.

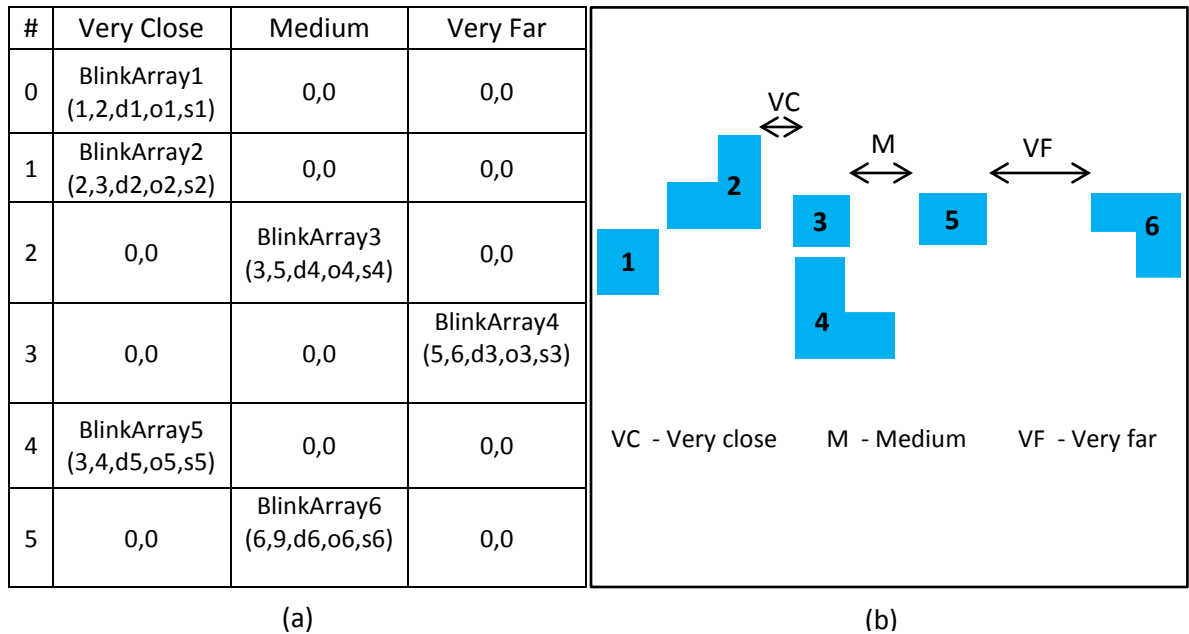


Figure 5.7 (a) 2D initial adjacency matrix based on the proximity hierarchy depending on the target map scale and (b) example of the distribution of buildings spaced at three levels - VC, M and VF. Outcome of clustering: buildings 1, 2, 3 and 4 are clustered as cluster ‘VC-1’, building 5 as ‘MD-0’ and building 6 as ‘VF-0’ (buildings 5 and 6 become single buildings).

vii. Based on the refined matrix, buildings are clustered on the proximity and clusters are filtered out for all the three levels. A separate field called cluster_id is added to the data set to store the cluster information. The cluster filtering algorithm is based on the identification of similar IDNs by comparing pairs that are linked across elements of each column (Figure 5.8) of the matrix and described as follows:

- a. Iterate through the elements of each of the three columns of the adjacency matrix other than elements with zero(s) in a pair.

- b. Get the linked pairs of building IDNs, finding a similar IDN of one pair in all the other pairs stored in elements of the particular column.
 - c. If similar building IDNs are found in two consecutive pairs or more, the cluster IDN of the first pair is assigned to other pairs.
 - d. If a building IDN of a pair in an element does not exist in all the pairs in other elements of the column, the next incremental cluster IDN is assigned to the available pair in the next element.
 - e. Go to the next element and follow steps (b) to (d) until the last row of the column.
- viii. Then classify clusters based on the proximity with the following labels:
- Clusters comprising of buildings in very close proximity are labelled as **VC-i** (where $i = 1, 2, \dots, n$).
 - Buildings which are very far are considered to be single clusters and labelled as **VF-0**.
 - A building, which gets isolated (single) such as building 5 (if one of the building IDNs in a pair is zero) in the medium proximity range, as described in step vi, is labelled as **MD-0**.
- ix. Buildings which belong to medium proximity clusters are stored in the memory for subsequent clustering process at the next hierarchical level (orientation difference).

#	Column	
0	26,25	-----> Cluster 1
1	27,26	
2	0,0	-----> Cluster 2
3	64,43	
4	0,0	
5	44,43	-----> Cluster 3
6	41,42	
7	0,0	
8	51,26	

Figure 5.8 An example of linked pairs and non-linked pairs with building IDNs in a column of the adjacency matrix. Each pair is stored in an element of the column.

- b) Clustering based on the orientation difference:
- i. Create a dynamic matrix to store building information related to the orientation difference (next constraint after proximity difference) using the clusters of buildings belonging to medium proximity except the single buildings that get isolated during clustering based on the proximity (e.g. a building with cluster label **MD-0**). The number of columns of the dynamic matrix depends on the number of linked pairs of buildings in a single medium cluster and the number of rows depends on the number of medium clusters formed during clustering based on the proximity.
 - ii. Iterate through each row of the dynamic matrix and check if all links in the connectivity list are available in the cluster of buildings in that row. If one or more links are missing, such links should be added to each cluster in the dynamic matrix with the help of the connectivity list. The missing links can be caused by the MST segmentation process with three proximity hierarchies - very close, medium and very far. Getting all new connectivity links between the building IDNs in one cluster is performed from the stored connectivity list generated during initial triangulation, and it is not necessary to run the DCT again within each cluster.
 - iii. However, such connectivity links must be refined in each cluster by excluding the link between each pair of buildings with the distance outside the medium distance range (distances within close range and very far range) if available.
 - iv. Run the MST for each cluster stored in each row in the dynamic matrix iteratively with the orientation difference as the weight.
 - v. Adopt similar steps iv to vii under Section (a) (clustering based on proximity) by creating the initial adjacency matrix for the two hierarchies of the orientation difference - small and very large - according to the threshold.

- vi. Classify clusters in the medium proximity range based on the orientation differences with the following labels:
 - Clusters comprising of buildings with large orientation differences are labelled as **ML-i** (where $i = 1, 2, \dots, n$).
 - A building, which gets isolated (single) due to a similar process described in step vi under Section (a) (clustering based on proximity), is labelled as **ML-0** (more priority is given to buildings in pairs with small orientation differences).
 - vii. Clusters comprising of buildings with small orientation differences are stored in the memory for subsequent clustering process at the next hierarchical level (similarity difference).
- c) Clustering based on the similarity difference index:
- i. This process is same as that described in the clustering steps i to iv under Section 5.2.2(b) to create the dynamic matrix using clusters comprising of buildings with small orientation differences. Next the two hierarchies of the similarity difference (next constraint after the orientation difference) - similar and dissimilar pairs of building geometries in shape - are used for clustering according to the threshold.
 - ii. Classify clusters in the medium proximity range based on the similarity difference index with the following labels:
 - Clusters comprising of buildings with similar shapes are labelled as **MS-i** (where $i = 1, 2, \dots, n$).
 - Clusters comprising of buildings with dissimilar shapes are labelled as **MDS-i** (where $i = 1, 2, \dots, n$).
 - A building, which gets isolated (single) due to a similar process described in step vi under Section (a) (clustering based on the proximity), is labelled as **MDS-0** (more priority is given to buildings in pairs that are similar in shape).

5.3 Testing of automatic clustering

A GUI using open source Java object-oriented programming language (see Appendix B.3) has been developed for automatic building clustering. Topographic data are stored in the open source PostgreSQL database supported by PostGIS spatial extension (PostGIS, 2013) to handle spatial data (see Appendix E.1 for the SQL queries used). The following three new fields are added to the implementation of the clustering process to the data stored in a spatial database.

- Global identification (*glbl_id*): This is used to identify each partitioned region in a data set uniquely according to the global constraints (contextual features).
- Local identification (*local_id*): This is used to assign each building in a particular region a unique IDN, which is used to find adjacency relationships of buildings using the DCT and the MST.
- Cluster identification (*cluster_id*): This is the field used to store cluster classification information about each building.

Initially, using Java GUI (frontend), the data stored in the spatial database (backend) are read to run the triangulation, to derive the Gestalt factors and to calculate the MST (Appendix B.3). Finally, the derived clusters are written to the database in the *cluster_id* field. In using the hierarchies of the Gestalt factors - proximity, orientation difference and the similarity difference - threshold values are to be used. The Java GUI enables setting these threshold values. Based on the experience as mentioned by Li *et al.* (2004), empirical value **0.5mm** of the target map scale is used as the separation threshold between two buildings. This is the threshold for the very close cluster in the proximity hierarchy. Threshold values for orientation and similarity are determined by the following criteria in this research:

Threshold values for the proximity:

- Very close distance threshold (minimum separation distance between two buildings): 0.5mm of the target map scale (Figure 5.7). Any two neighbourhood buildings which are spaced at a distance less than or equal to the above threshold value based on the target map scale are considered to be in the very close range of proximity in cluster formation.
- Medium distance threshold: $0.5\text{mm} < \mathbf{d} \leq 2\text{mm}$ of the target map scale. Any neighbouring buildings which are spaced at a distance \mathbf{d} are considered to be in the medium range of proximity.
- Very far distance threshold: $\mathbf{d} > 2\text{mm}$ of the target map scale. Any neighbouring buildings which are spaced at a distance \mathbf{d} are considered to be in the very far range of proximity.

Threshold value for the orientation difference:

- The orientation difference threshold is taken as the angle subtended in degrees by an arc of 0.125mm with a radius of 1m (Figure 5.9). Thus, the angle used is 7^0 degrees to generate results.

Threshold value for the similarity difference:

- The similarity difference index used in the generation of results is 0.75. Any value which is greater than or equal to 0.75 and less than or equal to 1 indicates that the two neighbourhood buildings are similar and any value which is less than 0.75 indicates that the two neighbourhood buildings are dissimilar in shape.

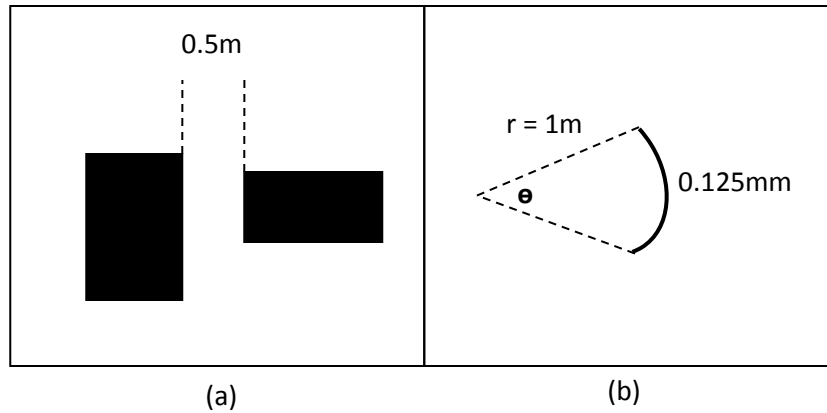


Figure 5.9 (a) Separation threshold between two buildings and (b) angle θ subtended by an arc of 0.125mm with radius $r = 1m$.

However, these threshold values for the medium and the very far ranges of proximity, orientation difference and the similarity difference are to be empirically determined.

5.3.1 Results of automatic clustering

Data set – I: Ordnance Survey MasterMap

Figure 5.10 represents an OS MasterMap source data set at the scale of 1 : 1.25K. The aim is to create a target map at the scale of 1 : 5K with the parameter values mentioned in Section 5.3 of the three Gestalt factors - proximity, orientation difference and the similarity difference. According to the clustering results in Figure 5.10 in a single region surrounded by the road network, it can be seen that there are fifteen (15) building clusters that are classified as 'very close' depending on the neighbourhood distance. There are seven (7) building clusters in the medium distance range, which satisfy the threshold for orientation difference but do not satisfy the threshold for the similarity difference index to become similar clusters. Buildings in these clusters are labelled as dissimilar. When observing the two dissimilar clusters in the medium range - MDS-6 and MDS-7 - buildings in both clusters satisfy the threshold for the orientation difference (i.e. orientation difference ≤ 7 degrees). However, the two buildings marked 'X' in both of these clusters as shown in Figure 5.10, which constitute a pair of buildings according to the neighbourhood relations in the triangulation, do not satisfy the threshold for the orientation difference. As a result, two clusters - each with a small orientation difference -

are formed. This is the reason as to why further clustering with the next constraint - the similarity difference - on these two clusters creates two dissimilar clusters labelled MDS-6 and MDS-7 in the clustering results. Therefore, this approach does not only serve as a potential method to identify buildings distributed in clusters, but also a method for so-called building alignments in clusters. There are five (5) single buildings that belong to the medium distance range according to cluster classification (i.e. buildings with cluster_id 'MD-0'). These are formed due to the consideration of hierarchical priority in forming the adjacency matrix for clustering (i.e. buildings that are very close are given more priority than those in the medium range). There is a single building that belongs to cluster_id 'ML-0'. When analysing this building and the building marked 'X' in cluster 'MDS-6', these two buildings belong to the medium range distance with a large orientation difference according to the threshold, but due to the hierarchical priority of orientation difference (i.e. 'small' and 'large'), the building in black colour gets isolated from the pair and classified as 'ML-0'. In the results, there are eight (8) buildings that are clustered as single entities with 'Very far' classification based on the very far distance threshold value.

It is important to mention that when changing the threshold values in the three Gestalt factors considered in the clustering depending on the target map scale, the results end up with different clusters. However, in order to use empirical values for proximity (medium range), orientation difference and similarity difference index, the automatic results need to be compared and validated with that of the manual cluster perception in the distribution of building geometries for subsequent automatic map generalization.



Figure 5.10 Automatic building clustering. Source data at the scale of 1 : 1.25K and the clusters are formed from the source data at the target map scale of 1 : 5K for the subsequent map generalization. Data source: OS MasterMap, Crown copyright.

Data set – II: Topographic data of the National Mapping Authority of Sri Lanka

Figure 5.11 depicts clustering results of building polygon geometries at the scale of 1 : 1K in three separate regions - R1, R2 and R3 - surrounded by the road network. The aim is to create a target map at the scale of 1 : 10K with the parameter values mentioned in Section 5.3 of the three Gestalt factors - proximity, orientation difference and the similarity difference. When observing the region R2, it should be noted that the algorithm for orientation by Duchêne *et al.* (2003) used in this research works well for the terrace shaped buildings.



Figure 5.11 Automatic building clustering on the source data at the scale of 1 : 1K, showing three different regions R1, R2 and R3 surrounded by the road network in (a), (b) and (c). Note: Each cluster is shown here by a unique colour.

The R1 region has only two clusters in which one belongs to very close proximity and the other belongs to a medium distance range with a larger orientation difference. Further, the R3 region has six (6) clusters with a close distance range, one cluster belonging to a medium distance range with similar shape and the other cluster in the medium distance range with dissimilar shape. All the other clusters are single buildings.

5.3.2 Synopsis of the contributing algorithms used

Table 5.2 summarises the contributing algorithms of the main polygon clustering algorithm developed in this research.

Table 5.2 Main polygon building clustering algorithm with the contributing algorithms: (1) new algorithm (Section 4.4) coupled with incremental point insertion based on the recursive edge-flipping technique by Berg *et al.* (2008) (2) wall statistical weighting for deriving building orientation by Duchêne *et al.* (2003) (3) new algorithm for similarity difference using the algorithm for discrete Hausdorff distance described by Atallah (1983) and Günter (1991) (Section 5.2.1) (4) MST (Prim, 1957) (5) new algorithm to segment the MST (Section 5.2.2) for creating clusters and (6) new algorithm for cluster classification (Section 5.2.2).

Main algorithm	#	Contributing algorithms	Purpose
Hierarchical building polygon clustering	1	DCT**	Topology and proximity derivation between buildings
	2	Wall statistical weighting*	Calculation of orientation difference between a building pair
	3	Similarity index measure using a combination of overlapping ratio, area to perimeter ratio (compactness) and discrete Hausdorff distance between a pair of buildings**	Similarity measure for shape and size between buildings in a pair
	4	MST*	To find near optimal neighbours in a tree with a cost value (e.g. distance)
	5	MST segmentation***	To identify hierarchical clusters
	6	Cluster classification***	To assign a sequential label to each cluster based on classification

* Existing algorithm

** Modification and/ or extension to an existing algorithm

*** New algorithm

5.3.3 Evaluation of the clustering results

The clustering results are evaluated based on the collected data with a two-phase focus group user testing that was carried out as explained in detail in Section 3.3.2.

Data collection - Phase I

The participants for this phase were chosen as described in Section 3.3.2. The focus group testing was conducted at the head office of the NMA of Sri Lanka on 10th August 2012 from 10.00 hrs to 12.00 hrs. In this phase, subjects were asked to apply manual clustering, following the instructions given in Appendix D.3 on a topographic map at the scale of 1 : 4K (Appendix D.1).

Procedure adopted for cluster evaluation:

- i. Assign clusters manually in each region (22 partitioned regions: see Appendix D.2) in the digital data set of source map stored in PostGIS based on the analogue results of the drawn clusters (see Appendices D.4 and D.5 for manual clustering results of randomly selected subjects from each group) collected using both expert and lay groups. There are 30 such feature layers created in PostGIS (layer names: exp1_poly to exp15_poly and lay1_poly to lay15_poly).
- ii. Create convex hulls of automatically generated clusters except single building clusters in PostGIS database based on the automatic clustering algorithm (feature layer names of the generated convex hulls: hull_glbl_i where i = 1-22).
- iii. Create a convex hull on each cluster created manually in the source data for each region by each subject from both lay and expert groups (feature layer names of the generated convex hulls: exp_i_glbl_j where i = 1, 15 and j = 1, 22 for expert group and lay_i_glbl_j, where i = 1, 15 and j = 1, 22 for lay group). See Appendix D.4 and D.5 for manual clustering results.
- iv. Then convex hulls under each region created by step (ii) above are matched with that of created by step (iii) by each subject using the symmetric difference of polygons with the JCS algorithm library (Vivid Solutions JCS, 2003) with a prototype developed in this

research (see Appendix B.4) as used by Revell and Antoine (2009). This matching is performed to understand how close the manual clustering results of subjects to the automatic clusters generated by the algorithm developed in this research. In this matching process, manual clusters which are matched to a value greater than or equal to 75% with that of automatic clusters are considered to be matching clusters.

- v. Finally, the evaluated results of automatic clustering and manual clustering in each region by the subjects in both groups are written to a file in ASCII format (see Appendix D.6 for the results in a tabular format).

Analysis of the clustering results - Phase I

Tables 5.3 and 5.4 summarise the clustering results of the expert and the lay groups respectively.

Table 5.3 Summary of the clustering results of the expert group derived from the cluster data given in Appendix D.6 in each of the twenty two regions compared with automatic clustering results. Highlights are the regions involved with hierarchical clustering.

Region IDN	No. of automatic clusters				Average no. of manual cluster classification by the group				Percentage cluster classified ratio (manual average/automatic)			
	VC	ML	MS	MDS	VC	ML	MS	MDS	VC	ML	MS	MDS
1	1	1	-	-	0.6	0	-	-	60	0	-	-
2	5	-	-	1	1	-	-	0	20	-	-	0
3	6	-	1	1	1.5	-	0	0	25	-	0	0
4	6	1	-	2	1.7	0	-	0	28	0	-	0
5	40	6	1	5	7.3	0	0	0	18	0	0	0
6	2	-	-	-	0.4	-	-	-	20	-	-	-
7	19	-	-	4	4.7	-	-	0	25	-	-	0
8	1	-	-	-	0.6	-	-	-	60	-	-	-
9	8	-	-	-	2	-	-	-	25	-	-	-
10	13	1	-	1	2.9	0	-	0	22	0	-	0
11	1	-	-	-	0.33	-	-	-	33	-	-	-
12	1	-	-	-	0.2	-	-	-	20	-	-	-
13	1	1	-	-	0.73	0	-	-	73	0	-	-
14	1	-	-	-	0.73	-	-	-	73	-	-	-
15	1	-	-	-	0.73	-	-	-	73	-	-	-
16	3	-	-	-	0.4	-	-	-	20	-	-	-
17	3	1	-	2	0.93	0	-	0	31	0	-	0
19	1	-	-	2	0.2	-	-	0.13	20	-	-	7
20	6	-	-	-	1.06	-	-	-	18	-	-	-
21	1	-	-	-	0.33	-	-	-	33	-	-	-
22	4	-	-	-	1.06	-	-	-	27	-	-	-

Table 5.4 Summary of the clustering results of the lay group derived from the cluster data given in Appendix D.6 in each of the twenty two regions compared with automatic clustering results. Highlights are the regions involved with hierarchical clustering.

Region IDN	No. of automatic clusters				Average no. of manual cluster classification by the group				Percentage cluster classified ratio (manual average/automatic)			
	VC	ML	MS	MDS	VC	ML	MS	MDS	VC	ML	MS	MDS
1	1	1	-	-	0.66	0	-	-	66	0	-	-
2	5	-	-	1	0.86	-	-	0	17	-	-	0
3	6	-	1	1	1.26	-	0	0	17	-	0	0
4	6	1	-	2	1.73	0	-	0	29	0	-	0
5	40	6	1	5	5.4	0	0	0	14	0	0	0
6	2	-	-	-	0.27	-	-	-	14	-	-	-
7	19	-	-	4	2.46	-	-	0	13	-	-	0
8	1	-	-	-	0.33	-	-	-	33	-	-	-
9	8	-	-	-	1.73	-	-	-	22	-	-	-
10	13	1	-	1	1.6	0	-	0	12	0	-	0
11	1	-	-	-	0.33	-	-	-	33	-	-	-
12	1	-	-	-	0.13	-	-	-	13	-	-	-
13	1	1	-	-	0.27	0	-	-	27	0	-	-
14	1	-	-	-	1	-	-	-	100	-	-	-
15	1	-	-	-	0.87	-	-	-	87	-	-	-
16	3	-	-	-	0.13	-	-	-	4	-	-	-
17	3	1	-	2	0.4	0	-	0	13	0	-	0
19	1	-	-	2	0	-	-	0	0	-	-	0
20	6	-	-	-	0.4	-	-	-	7	-	-	-
21	1	-	-	-	0.13	-	-	-	13	-	-	-
22	4	-	-	-	0.93	-	-	-	23	-	-	-

When observing the summarised results in Tables 5.3 and 5.4, it can be seen that both lay and expert groups have attempted clustering of buildings belonging to very close range distance, although results are not so satisfactory on average in identifying such clusters in all the regions. When comparing the results of both groups, regions with a few or single clusters of buildings belonging to very close range (regions 1, 13, 14 and 15 in Figure 5.12) have shown a rather satisfactory cluster classification ratio.

The reason is that the buildings in these regions have been clustered based on proximity alone without hierarchical clustering perception which involves partitioning clusters into multiple levels as described in Section 2.3.2. However, similar types of clusters in regions 11, 12 and 16 show a low percentage in the Tables 5.3 and 5.4. This could be due to the

136

The lay group has perceived clusters of buildings in the medium distance range more satisfactorily than the expert group when considering the average percentages of the subjects who recognized manual clusters belonging to the medium range in both expert and lay groups in all the partitioned regions where hierarchical clustering is involved (lay group: 9.3% and expert group: 8%, derived from the results of Table 5.5).

Table 5.5 Summary of the results in identifying the medium distance range clusters for the hierarchical application of the Gestalt constraints - orientation and similarity difference - by the expert and the lay groups, derived from the cluster data given in Appendix D.6.

Region IDN involved hierarchical clustering	Percentage of subjects who recognized manual clusters belonging to medium distance range	
	Expert group	Lay group
2	0	13
3	40	13
4	0	7
5	13	13
7	0	7
10	7	7
13	0	7
17	7	0
19	20	27
21	0	7

It is also understood that the results of correctly classified clusters in regions with larger areas (see regions 5 and 7 in Figure 5.13(a) delineated in brown colour - see Appendix D.2 for a more enlarged view) by both groups are very poor. This could be further emphasised when analysing the percentages of correctly identified clusters with a misclassification in regions 5 and 7 by the subjects in both expert and lay groups in Table 5.6 below. The reason could be that the subjects have lost concentration while making an strenuous effort to perceive and categorise clusters belonging to different types of classifications during the manual hierarchical clustering. The subjects performed this clustering as instructed in the experiment (see Appendix D.3) using the cluster classification and labelling method in the clustering algorithm explained in Section 5.2.2, implemented for automatic clustering in this research.

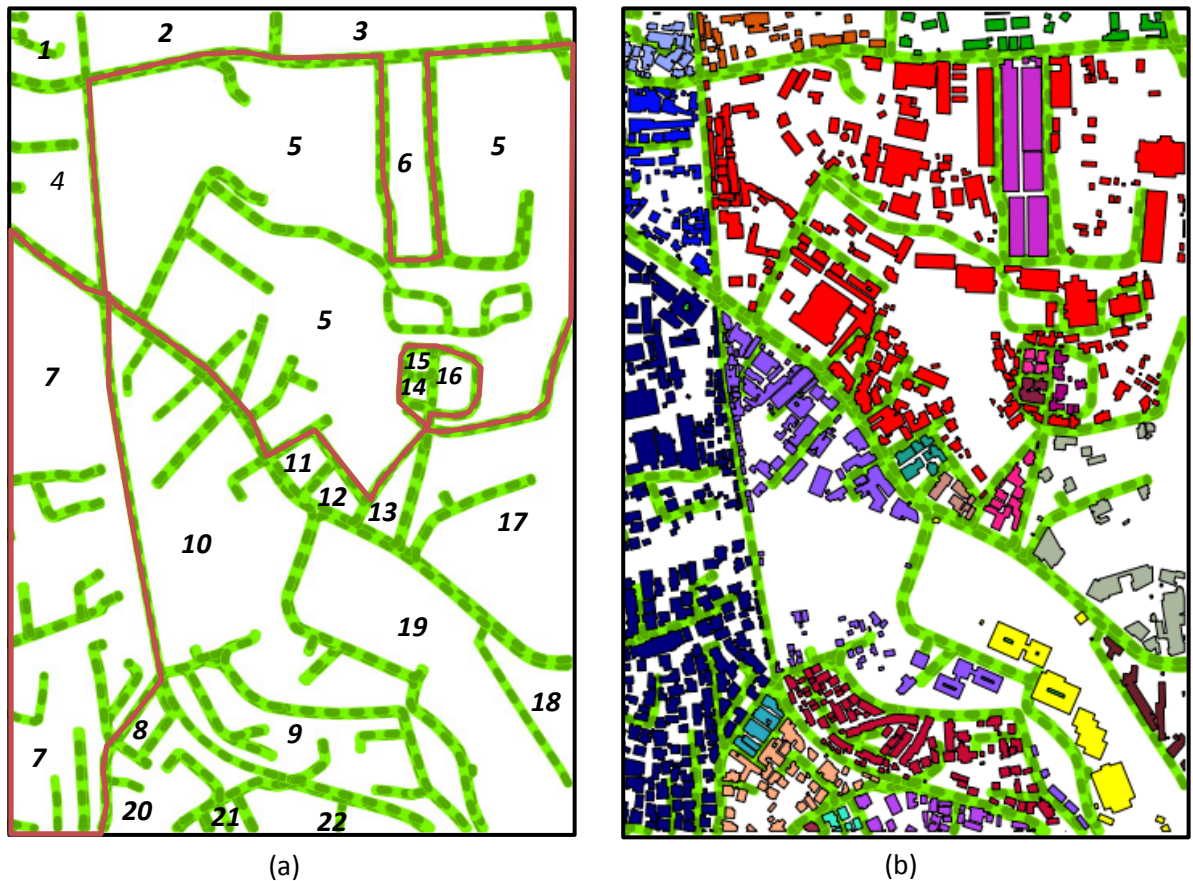


Figure 5.13 (a) Partitioned regions surrounded by the road network where inner roads are ignored and (b) building features within each region.

When considering the results of clustering by the two groups shown in Tables 5.3 and 5.4 above, the lay group has not performed hierarchical clustering with the classification in any of the regions, while the expert group has performed such clustering only in region 19, but with a very low cluster classification ratio. The clear, typical building configuration in region 19 should have enabled them to perceive hierarchical clusters (Figure 5.12(d)). However, the lay group has got the highest percentage rate of identifying buildings in the medium distance range in region 19 when observing Table 5.5 which gives information about the regions in which hierarchical clustering was attempted by the participants. This emphasises that the lay group has at least attempted hierarchical clustering. This appears to arise because it is the buildings in the medium range that are hierarchically clustered after clustering buildings in the very close regions with the proximity in the next step based on the orientation and the similarity as discussed in the clustering algorithm in Section 5.2.2.

Table 5.6 Summary of the results of percentages of the subjects in both groups identifying clusters with a misclassification, derived from the cluster data given in Appendix D.6.

Region IDN	Percentages of correctly identified clusters manually in each region, but with a misclassification	
	Expert group	Lay group
1	-	7
2	40	40
3	53	20
4	40	33
5	93	80
6	13	27
7	87	47
8	20	60
9	27	13
10	67	53
11	13	13
12	7	20
13	20	40
14	-	-
15	-	7
16	27	13
17	60	27
18	13	-
19	40	7
20	33	27
21	7	13
22	20	20

Data collection - Phase II

The same participants who took part in the first phase of user testing were used in this phase as well. This user testing was also conducted at the head office of the NMA of Sri Lanka on 13th September 2012 from 09.30 hrs to 10.30 hrs, almost one month after the elapse of phase I. In this phase, the idea was to evaluate the automatic clustering approach with the feedback of subjects by way of a structured questionnaire (see Appendix D.7), comparing each of their manual clustering results with that of the automatic results as explained in Section 3.3.2.

Analysis of the clustering results – Phase II

In the analysis of the results, Table 5.7 summarises the answers of questions 1 and 2, Table 5.8 for questions 5(a), 6(a) and 7(a), Table 5.9 for questions 3(a), 8 and 9(a), and Table 5.10 for questions 4 and 10 posed in the structured questionnaire (see Appendix D.7).

Table 5.7 Evaluation of the automatic clustering method by the expert and the lay groups.

Type of analysis	Results	Outcome															
(i). Rating cluster result	<table border="1"> <caption>Data for (i). Rating cluster result</caption> <thead> <tr> <th>Rating</th> <th>Expert group (%)</th> <th>Lay Group (%)</th> </tr> </thead> <tbody> <tr> <td>Poor</td> <td>0</td> <td>0</td> </tr> <tr> <td>Fair</td> <td>13</td> <td>13</td> </tr> <tr> <td>Good</td> <td>73</td> <td>80</td> </tr> <tr> <td>Excellent</td> <td>13</td> <td>7</td> </tr> </tbody> </table>	Rating	Expert group (%)	Lay Group (%)	Poor	0	0	Fair	13	13	Good	73	80	Excellent	13	7	87% of the subjects in both groups rated automatic clustering good to Excellent
Rating	Expert group (%)	Lay Group (%)															
Poor	0	0															
Fair	13	13															
Good	73	80															
Excellent	13	7															
(ii). Usefulness of cluster classification for map generalization	<table border="1"> <caption>Data for (ii). Usefulness of cluster classification for map generalization</caption> <thead> <tr> <th>Response</th> <th>Expert group (%)</th> <th>Lay Group (%)</th> </tr> </thead> <tbody> <tr> <td>Strongly disagree</td> <td>0</td> <td>0</td> </tr> <tr> <td>Disagree</td> <td>0</td> <td>0</td> </tr> <tr> <td>Agree</td> <td>60</td> <td>73</td> </tr> <tr> <td>Strongly agree</td> <td>40</td> <td>27</td> </tr> </tbody> </table>	Response	Expert group (%)	Lay Group (%)	Strongly disagree	0	0	Disagree	0	0	Agree	60	73	Strongly agree	40	27	100% of subjects in both groups agreed
Response	Expert group (%)	Lay Group (%)															
Strongly disagree	0	0															
Disagree	0	0															
Agree	60	73															
Strongly agree	40	27															

Table 5.8 Evaluation of the use of threshold values of the Gestalt factors - proximity, orientation and similarity in the automatic clustering method by the expert and the lay groups.

Type of analysis	Results	Outcome															
(i). Compliance with the proximity (medium range - $0.5\text{mm} \leq D \leq 2\text{mm}$)	<table border="1"> <caption>Data for Proximity Compliance</caption> <thead> <tr> <th>Response</th> <th>Expert group (%)</th> <th>Lay Group (%)</th> </tr> </thead> <tbody> <tr> <td>Strongly disagree</td> <td>0</td> <td>0</td> </tr> <tr> <td>Disagree</td> <td>7</td> <td>7</td> </tr> <tr> <td>Agree</td> <td>93</td> <td>93</td> </tr> <tr> <td>Strongly agree</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Response	Expert group (%)	Lay Group (%)	Strongly disagree	0	0	Disagree	7	7	Agree	93	93	Strongly agree	0	0	93% of subjects in both groups agreed
Response	Expert group (%)	Lay Group (%)															
Strongly disagree	0	0															
Disagree	7	7															
Agree	93	93															
Strongly agree	0	0															
(ii). Compliance with the orientation difference threshold (7° degrees)	<table border="1"> <caption>Data for Orientation Compliance</caption> <thead> <tr> <th>Response</th> <th>Expert group (%)</th> <th>Lay Group (%)</th> </tr> </thead> <tbody> <tr> <td>No</td> <td>27</td> <td>33</td> </tr> <tr> <td>Yes</td> <td>67</td> <td>67</td> </tr> <tr> <td>No response</td> <td>7</td> <td>0</td> </tr> </tbody> </table>	Response	Expert group (%)	Lay Group (%)	No	27	33	Yes	67	67	No response	7	0	<ul style="list-style-type: none"> - 67% of subjects in both groups agreed - Average of 30% of the subjects in both groups disagreed 			
Response	Expert group (%)	Lay Group (%)															
No	27	33															
Yes	67	67															
No response	7	0															
(iii). Compliance with the similarity difference index (0.25)	<table border="1"> <caption>Data for Similarity Compliance</caption> <thead> <tr> <th>Group</th> <th>No (%)</th> <th>Yes (%)</th> </tr> </thead> <tbody> <tr> <td>Expert group</td> <td>0</td> <td>100</td> </tr> <tr> <td>Lay group</td> <td>0</td> <td>100</td> </tr> </tbody> </table>	Group	No (%)	Yes (%)	Expert group	0	100	Lay group	0	100	All subjects in both groups agreed						
Group	No (%)	Yes (%)															
Expert group	0	100															
Lay group	0	100															

Table 5.9 Evaluation of the comparison of manual clustering approach with the automatic clustering by the expert and the lay groups.

Type of analysis	Results	Outcome															
(i). Compliance of manual clustering results with that of automatic clustering	<table border="1"> <caption>Data for (i) Compliance of manual clustering results with that of automatic clustering</caption> <thead> <tr> <th>Category</th> <th>Expert group (%)</th> <th>Lay group (%)</th> </tr> </thead> <tbody> <tr> <td>Strongly mismatch</td> <td>8</td> <td>0</td> </tr> <tr> <td>Mismatch</td> <td>15</td> <td>0</td> </tr> <tr> <td>Match</td> <td>80</td> <td>100</td> </tr> <tr> <td>Strongly match</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Category	Expert group (%)	Lay group (%)	Strongly mismatch	8	0	Mismatch	15	0	Match	80	100	Strongly match	0	0	<ul style="list-style-type: none"> - Lay group claims that their results are totally matched (100%) - Expert group claims that their match is about 80%
Category	Expert group (%)	Lay group (%)															
Strongly mismatch	8	0															
Mismatch	15	0															
Match	80	100															
Strongly match	0	0															
(ii). Adaptation of hierarchical clustering in the manual clustering approach	<table border="1"> <caption>Data for (ii) Adaptation of hierarchical clustering in the manual clustering approach</caption> <thead> <tr> <th>Category</th> <th>Expert group (%)</th> <th>Lay group (%)</th> </tr> </thead> <tbody> <tr> <td>No</td> <td>7</td> <td>7</td> </tr> <tr> <td>Yes</td> <td>93</td> <td>87</td> </tr> <tr> <td>No response</td> <td>0</td> <td>7</td> </tr> </tbody> </table>	Category	Expert group (%)	Lay group (%)	No	7	7	Yes	93	87	No response	0	7	<p>Majority of both groups (more than 87%) claim that they applied the concept of hierarchical clustering</p>			
Category	Expert group (%)	Lay group (%)															
No	7	7															
Yes	93	87															
No response	0	7															
(iii). Significance of difference between manual cluster perception and automatic clustering when comparing results (manual vs. automatic)	<table border="1"> <caption>Data for (iii) Significance of difference between manual cluster perception and automatic clustering when comparing results (manual vs. automatic)</caption> <thead> <tr> <th>Category</th> <th>Expert group (%)</th> <th>Lay group (%)</th> </tr> </thead> <tbody> <tr> <td>No</td> <td>40</td> <td>53</td> </tr> <tr> <td>Yes</td> <td>60</td> <td>40</td> </tr> <tr> <td>No response</td> <td>0</td> <td>7</td> </tr> </tbody> </table>	Category	Expert group (%)	Lay group (%)	No	40	53	Yes	60	40	No response	0	7	<ul style="list-style-type: none"> - Majority (53%) in the lay group claim no significance difference - Majority (60%) in the expert group claim a significant difference 			
Category	Expert group (%)	Lay group (%)															
No	40	53															
Yes	60	40															
No response	0	7															

Table 5.10 Evaluation of the adaptation of manual clustering process by the expert and the lay groups.

Type of analysis	Results	Outcome																		
(i). Application of the algorithm manually with hierarchical perception of Gestalt factors	<table border="1"> <caption>Data for (i). Application of the algorithm manually with hierarchical perception of Gestalt factors</caption> <thead> <tr> <th>Category</th> <th>Expert group (%)</th> <th>Lay group (%)</th> </tr> </thead> <tbody> <tr> <td>Very difficult</td> <td>7</td> <td>0</td> </tr> <tr> <td>Difficult</td> <td>40</td> <td>40</td> </tr> <tr> <td>Easy</td> <td>47</td> <td>53</td> </tr> <tr> <td>Very easy</td> <td>7</td> <td>7</td> </tr> </tbody> </table>	Category	Expert group (%)	Lay group (%)	Very difficult	7	0	Difficult	40	40	Easy	47	53	Very easy	7	7	<p>- 47% of subjects in the expert group claim it to be difficult to very difficult</p> <p>- 40% of subjects in the lay group claim it difficult</p>			
Category	Expert group (%)	Lay group (%)																		
Very difficult	7	0																		
Difficult	40	40																		
Easy	47	53																		
Very easy	7	7																		
(ii). Overall experience in the hierarchical clustering approach	<table border="1"> <caption>Data for (ii). Overall experience in the hierarchical clustering approach</caption> <thead> <tr> <th>Category</th> <th>Expert group (%)</th> <th>Lay group (%)</th> </tr> </thead> <tbody> <tr> <td>Highly unsatisfactory</td> <td>7</td> <td>0</td> </tr> <tr> <td>Unsatisfactory</td> <td>13</td> <td>27</td> </tr> <tr> <td>Neutral</td> <td>13</td> <td>13</td> </tr> <tr> <td>Satisfactory</td> <td>67</td> <td>53</td> </tr> <tr> <td>No response</td> <td>0</td> <td>7</td> </tr> </tbody> </table>	Category	Expert group (%)	Lay group (%)	Highly unsatisfactory	7	0	Unsatisfactory	13	27	Neutral	13	13	Satisfactory	67	53	No response	0	7	<p>- 67% of subjects in the expert group claim it to be satisfactory</p> <p>- 53% of subjects in the lay group claim it to be satisfactory</p>
Category	Expert group (%)	Lay group (%)																		
Highly unsatisfactory	7	0																		
Unsatisfactory	13	27																		
Neutral	13	13																		
Satisfactory	67	53																		
No response	0	7																		

Table 5.11 Summary of the answers to the Question 3(b) of the questionnaire in Appendix D.7.

Reasons for cluster mismatch	Expert group	Lay group
Varying perception from subject to subject	✓	✓
Adherence to a specific criteria by the automatic method	✓	✗
Difficulty in distinguishing medium distance range	✓	✗
Difficulty in differentiating orientation difference between buildings	✗	✓

According to the results of Table 5.11, both groups have mentioned that the cluster mismatch is mainly due to the varying perception from subject to subject in addition to the other reasons given by each of the two groups.

Table 5.12 Summary of the answers to the Question 4(b) of the questionnaire in Appendix D.7.

Reasons for the difficulty in applying the Gestalt factors hierarchically	Expert group	Lay group
Application of orientation and similarity in shape and size is difficult	✓	✓
Keeping a lot of information in mind (cognitive load high) difficult to handle	✓	✗
Manual process time consuming	✓	✗
Requires further experience	✓	✗
Difficulty in referring to target result every time in the manual clustering process	✗	✓
Thresholds applied are guessed in manual method, but in automatic method, absolute values are used	✗	✓

Table 5.12 elaborates the results of difficulties encountered by the subjects in applying the Gestalt factors manually in a hierarchical manner depicted in Table 5.10(i). According to the results of Table 5.12, both groups have concluded that the hierarchical approach of applying the Gestalt factors - orientation and similarity in shape and size - are difficult. Another important reason for the difficulty mentioned is the cognitive load which subjects have to undergo in manual clustering as mentioned by the expert subjects.

7% of the subjects of both groups have disagreed with the medium distance range (md) used in the automatic clustering method as depicted in Table 5.8(i) related to Question 5(b) in the questionnaire. Subjects of the expert group were of the view that the range should be within 0.5mm and 4mm ($0.5\text{mm} \leq \text{md} \leq 4\text{mm}$) while the lay group was of the view that it should be within 1mm and 5mm ($1\text{mm} \leq \text{md} \leq 5\text{mm}$). However, the lower threshold value (1mm) suggested by the lay group cannot be used since the empirical value 0.5mm has been the value requested to be used in the manual clustering approach during the experimentation (see Appendix D.3).

Table 5.13 Summary of the answers to the Question 6(b) of the questionnaire in Appendix D.7.

Reasons for the non-compliance with the orientation threshold	Expert group	Lay group
Arc distance used in the algorithm smaller (use arc distance of 2mm or greater and check results)	✓	✗
For small buildings, with 7 ⁰ degrees, no significant difference of orientation between pairs of buildings observed (smaller the building, larger the threshold value to be applied - requires further testing)	✗	✓

Table 5.13 elaborates the results of non-compliance of the orientation threshold used by the subjects with that used in the automatic method depicted in Table 5.7(ii). When analysing the reasons given by the subjects in Table 5.13, further testing of the orientation threshold needs to be carried out.

Table 5.14 Summary of the answers to the Question 9(b) of the questionnaire in Appendix D.7.

Reasons for the significant difference	Expert group	Lay group
Non-consideration of hierarchical clustering	✓	✗
Change of clustering results due to varying human perception	✓	✓
Distance criteria applied was different	✗	✓

Table 5.14 elaborates the results of the significant difference between the manual cluster perception and the automatic clustering depicted in Table 5.9(iii) above. When analysing the reasons given by the subjects in Table 5.14, the main reason given by both groups is the varying perception of clusters from subject to subject.

Implications of the cluster evaluation

Based on the analysis of the results of phase I and phase II of the manual clustering approach in comparison with the automatic clustering method, it can be understood from the results that none of the groups have perceived hierarchical clustering as done by the automatic clustering method developed in this research. However, the subjects have perceived clusters based on proximity rather than the other two Gestalt factors - orientation difference and the similarity difference in shape and size - where no hierarchical clustering is involved. When considering the clusters that involve hierarchical clustering with proximity followed by the orientation difference and the similarity difference in shape and size, their ability to distinguish such clusters is very poor. Among the main reasons for this inability that can be drawn from the analysis are: (a) high cognitive load in the application of Gestalt factors hierarchically in regions within which a large numbers of clusters exist (b) less capability in distinguishing orientation difference between an adjacent pair of buildings, especially when one building is very small compared to the other building in the pair and (c) human perception differences in guessing threshold regions in the application of the Gestalt factors. However, the subjects have not found it difficult to distinguish the similarity difference between a pair of buildings compared to the orientation.

It is also important to mention that one of the intentions of the experiment is to get an insight into the use of threshold regions in the application of the three Gestalt factors - proximity, orientation difference, and similarity difference in shape and size - for the purpose of deriving generalized maps at the target scale of 1 : 10K only. Depending on the scale and the purpose of the generalized map, these threshold regions can vary.

In the whole evaluation process, the clusters were treated within a region surrounded by the contextual features (regions surrounded by the road network in the experiment). Subjects were instructed to ignore inner roads within each region on the assumption that they would be removed in the generalization process (see Appendix D.3). But in reality, there can be inner roads that should be represented in the target generalized map. When visualising automatic clustering results, it should be realised that the incorporation of

inner contextual features is necessary for separating clusters more appropriately for subsequent generalization. This can be further emphasised when observing the very close cluster labelled 'VC-1' in Figure 5.14. Since building adjacency relationships within the region have been considered ignoring inner roads, cluster - 'VC-1' - has got buildings on either side of the inner road.



Figure 5.14 Automatic building clusters (yellow) in region 9 used in the clustering experiment.

In order to consider inner contextual features of buildings within a region, not only the adjacency relationships between building features, but also the adjacency relationships between buildings and other contextual features inside the region must be derived and used in the clustering process. The DCT developed in this research and used in the clustering process deals only with polygon geometries. When contextual features such as roads and hydrographic features are considered, both line and polygons of such geometries exist and, therefore, the triangulation algorithm used should be able to handle both polygon and line geometries.

Modification to the clustering algorithm

In order to enable the derivation of adjacency relationships between buildings, taking into account contextual features, the clustering approach is further improved with the extension described in Section 4.2.2 to attach feature geometry IDNs of buildings and contextual features to Steiner points introduced in the triangles generated by the CNDT described in Section 4.2 in a new data enrichment prototype (see Appendix B.5). In this approach, line geometries of contextual features - roads and hydrographic features - are processed together with building polygons to derive adjacency relationships as shown in Figure 5.15. Hence the DCT method (see Table 5.2) is replaced with the modified CNDT in the automatic clustering approach. However, the DCT that can only deal with polygons will be used in the generalization phase for retrieving adjacency relationships between buildings.

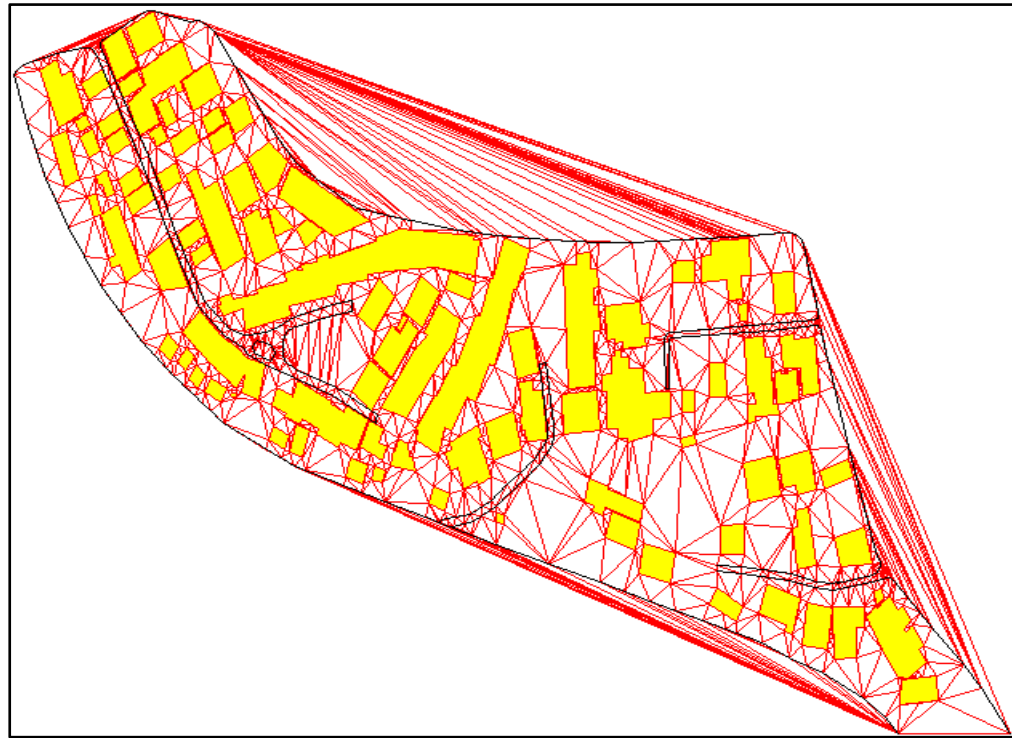


Figure 5.15 Generation of the CNDT using buildings and roads with the enforcement of their edges as constraints within region 9 used in the clustering experiment.

Figure 5.16 depicts the results of the automatic clustering within region 9 after incorporating contextual roads inside the region in the automatic clustering process. Now it can be observed that the cluster labelled 'VC-1' (see Figure 5.14) obtained by the previous automatic clustering without taking into account the contextual inner roads, is now split into two on either side of the road when observing the results.



Figure 5.16 Automatic building clusters (light green) in region 9 after deriving adjacency relationships between buildings, taking into account the contextual inner roads within the region.

There can also be rare instances where some buildings get isolated depending on their location in relation to contextual features (e.g. a single building gets isolated from other buildings inside an inner circular road). If such buildings exist, these buildings are not considered in the automatic clustering process because they are not captured in creating the building adjacency list in the triangulation. As a result, such buildings are considered to be non-clustered (not present in the initial adjacency matrix) and assigned a label 'NC-0' in addition to the cluster classification described in the automatic clustering algorithm in Section 5.2.2.

5.4 Shape enrichment of clusters

Once the automatic clustering is complete, some more enrichment to each cluster should be assigned in order to support choice of building aggregation algorithms based on the shape of the outline (orthogonal or non-orthogonal) of building polygons that touch the outline of each cluster polygon. The reason is that after the aggregation of buildings in clusters during the subsequent automatic generalization process, the shape of the outline and orientation of the aggregated building needs to be maintained depending on the shape of the outline and the orientation of buildings of the cluster. The clusters of the buildings touching the outline with orthogonal sides have to be aggregated maintaining orthogonality of the sides of the new aggregated buildings while buildings with non-orthogonal sides on the outline can be aggregated without maintaining orthogonality. Although the orientation has already been dealt with in the clustering process, its impact on the shape of the outline of buildings in a cluster has not been considered. Therefore, the enrichment of clusters in relation to the shape of their outline for determining the orientation is an important criterion and will be dealt with in this Section.

5.4.1 Testing of algorithms for retrieving buildings at the cluster outline

Convex Hull based algorithm

- (i) Create convex hull of the union of buildings in each cluster iteratively.
- (ii) Iterate through each building in the union and identify buildings that share an edge with the convex hull.
- (iii) Go to the next cluster and follow steps (i) and (ii) until the end of all clusters.

Analysis of the results

When analysing the buildings that share a side with the convex hull in the cluster in Figure 5.17(a), the building IDN 7 that potentially should touch the cluster outline is not chosen due to its significant concavity in shape. However, when the cluster shape is convex, convex hull approach works well (Figure 5.17(b)) and it chooses all the buildings (e.g.

buildings with IDNs 3, 4, 6, 7 and 8) that can potentially touch the cluster outline. Therefore, using convex hull approach in determining the shape of clusters is not a suitable approach for finding buildings that touch the cluster outline.

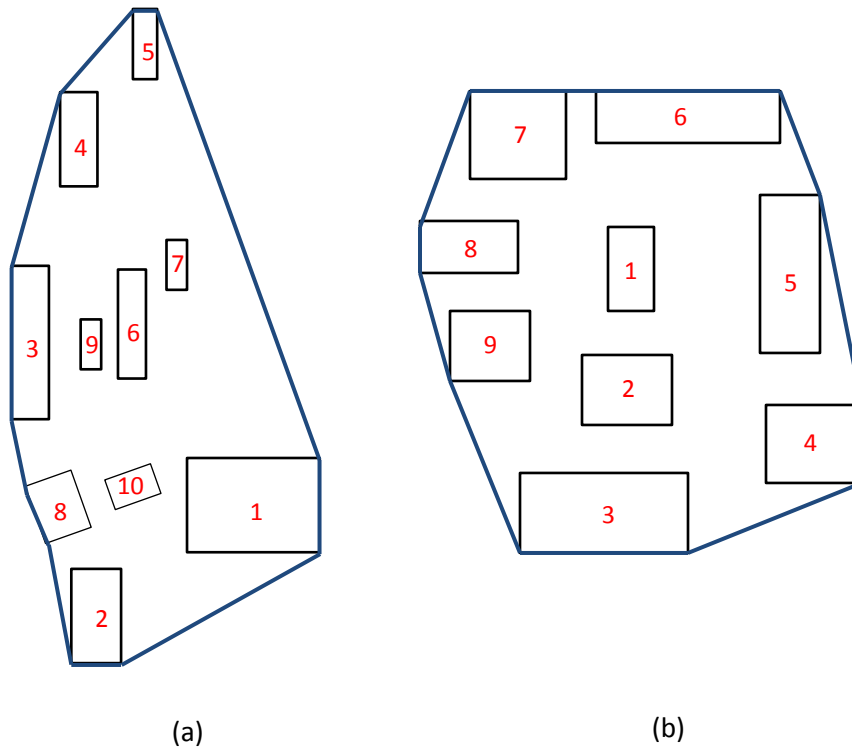


Figure 5.17 Building clusters on synthetic data: (a) convex hull of the cluster of concave shape and (b) cluster of convex shape.

Concave hull based algorithm

For this purpose, the open source Java library for concave hull implementation with JTS (Concave hull, 2013) based on the algorithm developed by Duckham et al. (2008) is used.

- (i) Create concave hull of the union of buildings in each cluster iteratively.
- (ii) Iterate through each building in the union and identify buildings that share an edge with the concave hull.
- (iii) Go to the next cluster and follow steps (i) and (ii) until the end of all clusters.

Analysis of the results

When testing the implementation with synthetic data, it is realised that the results are a little bit unexpected when the algorithm is used with different distance thresholds (see (a) and (b) in Figures 5.18 and 5.19). However, when a distance slightly larger than the maximum side length of all the sides of buildings in the cluster is used, the concave hull generated is quite satisfactory for identifying border touching buildings in both convex and concave clusters (see (c) in Figures 5.18 and 5.19). A problem may occur in finding the maximum side length of all the buildings if the edges have splits at closer intervals in the data set (building IDN 1 in Figure 5.19 has edges split into short segments). Therefore, it is required to simplify each building geometry to remove such splits before finding out the maximum length to be used in generating the concave hull. The next sub-topic will introduce an improved concave hull based algorithm in this research to extract the buildings that touch the outline of the cluster.

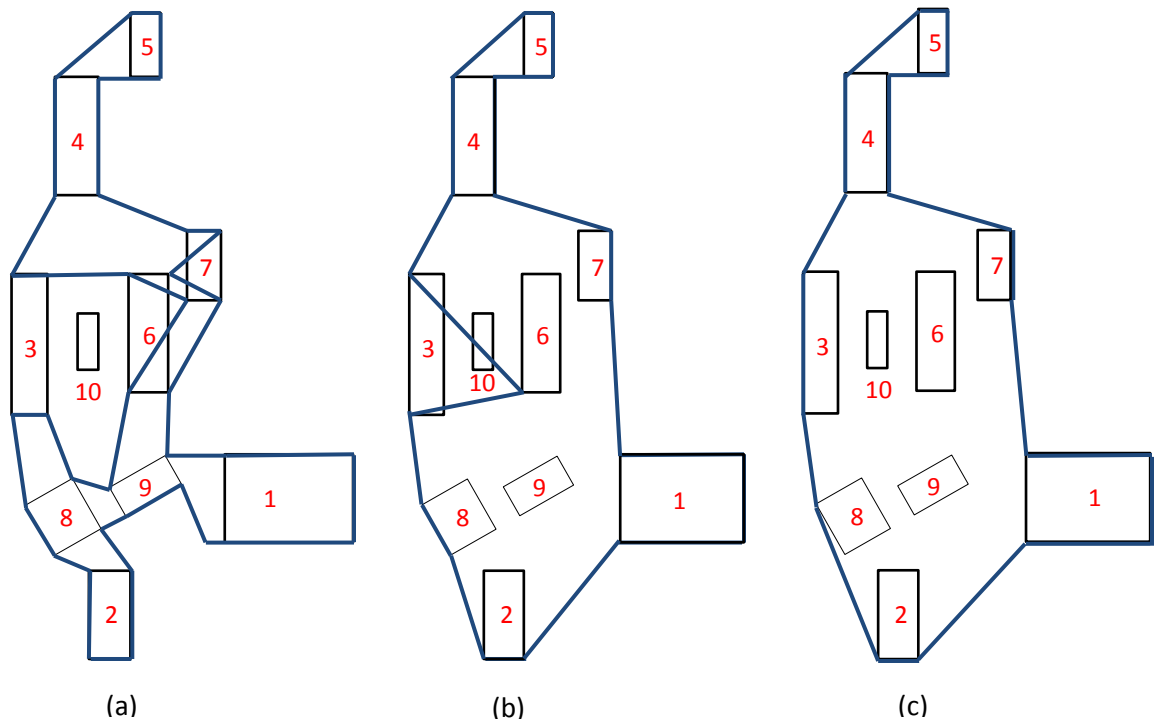


Figure 5.18 Concave hull generation of a concave cluster of buildings on synthetic data: (a) concave hull with 0.0 m distance threshold (b) with 10.0 m distance threshold and (c) with 12.1 m distance threshold which is slightly greater than the maximum distance of all the building edges in the cluster.

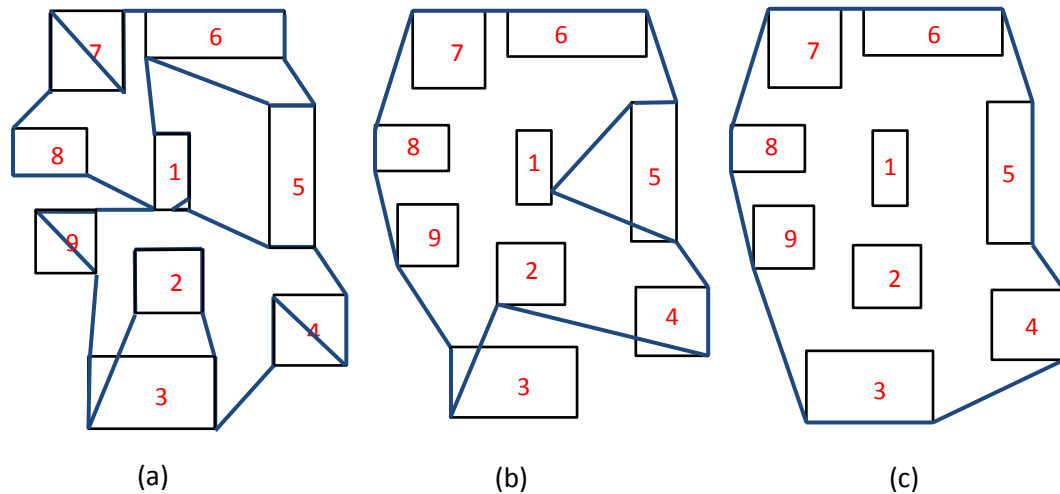


Figure 5.19 Concave hull generation of a convex cluster of buildings on synthetic data: (a) concave hull with 0.0 m distance threshold (b) with 10.0 m distance threshold and (c) with 11.56 m distance threshold which is slightly greater than the maximum distance of all building edges in the cluster.

Improved Concave hull based algorithm

- i. Iterate through each building geometry in the cluster and get a building.
- ii. Iterate through each connected pair of edges of the building and find the common vertex between the pair.
- iii. Work out the angle formed by the pair at the common vertex. If the angle formed $\leq 1^\circ$, the vertex is a candidate to be removed from the building geometry.
- iv. Check if the number of edges > 3 . If yes, remove the candidate vertex from the geometry.
- v. If the number of edges = 3, the geometry becomes a triangle, and the removal of a vertex will lead to an invalid geometry polygon. Therefore, when the geometry is a triangle, simplification is stopped.
- vi. After all the edge iterations, what is left is a collection of simplified vertices. Recreate the simplified geometry using this collection of vertices.

- vii. Iterate through each side of the simplified geometry and find out the maximum side length.
- viii. Generate the concave hull with a distance threshold slightly greater than the maximum side length (e.g. distance threshold = maximum side length + 0.1).

5.4.2 Testing of new algorithms for the cluster shape enrichment

This Section describes how a new algorithm has been developed, tested and refined in this research based on the improved concave hull generation algorithm as explained in the previous Section 5.4.1. Clusters belonging to the larger orientation category (i.e. clusters starting with the label 'ML' as described in Section 5.2.2 above) and isolated buildings are not subject to the shape enrichment, hence not considered in the query. Each cluster of buildings with a larger orientation difference is recorded as non-orthogonal while each cluster of buildings with a smaller orientation difference is recorded as orthogonal in the spatial database. The algorithm has been implemented in the spatial clustering GUI (see Appendix B.5). SQL queries used are given in Appendix E.1.

Cluster shape enrichment algorithm I

- i. Query clusters belonging to a very close region (i.e. labels starting with 'VC') and clusters in the medium range with a smaller orientation difference (i.e. labels starting with 'MS' and 'MDS') and create a union of buildings with the group cluster IDN and its classification iteratively.
- ii. In each cluster, create a concave hull of the union of buildings based on the improved concave hull based algorithm given above.
- iii. Iterate through each building in the union and identify buildings that share either an edge or point with the concave hull (the buildings that touch the outline of each cluster).
- iv. Check orthogonality of each such building in the building union (cluster) with the building orientation algorithm by Duchêne *et al.* (2003).

- v. If all the buildings that touch the outline of the cluster have orthogonal sides i.e. confidence interval $\geq 80\%$ as defined by Duchêne *et al.* (2003), record shape of such clusters to be orthogonal in the spatial database layer.
- vi. If confidence interval $< 80\%$, record the particular cluster as non-orthogonal in the spatial database layer.
- vii. Go to the next cluster and perform the steps (ii) to (vi) until the end of all clusters.

Analysis of the results of the shape enrichment algorithm I

When analysing the cluster shape enrichment results in Figure 5.20, it is observed that some very close clusters identified to be orthogonal by the cluster shape enrichment algorithm, do not possess a reasonable orthogonal outline (e.g. Clusters labelled VC-6 in regions 4 and 5, and clusters labelled VC-30 and VC-31 in region 5 delineated in red colour).



Figure 5.20 Shape enrichment of clusters in three regions 4, 5 and 7 (part of) surrounded by the road network: clusters of orthogonal shape are shown in cyan colour while clusters of non-orthogonal shape are shown in yellow colour with the cluster label of each building shown in the cluster. Data source: 1 : 1K data of the NMA (Sri Lanka). Copyright reserved.

When analysing the very close clusters, it is understood that not only the orthogonality of buildings, but also the orientation of buildings in the cluster outline makes a significant effect in determining the orthogonal shape of a cluster. This may have happened because during the clustering process, only the proximity Gestalt factor is used in creating a very close cluster, assuming that there is no any significant orientation difference between buildings located very close in such clusters. However, this type of exceptional case can occur in different data sets. Therefore, further modification to the cluster shape enrichment algorithm I is required and dealt with next.

Cluster shape enrichment algorithm II

In this algorithm, in addition to finding orthogonality of buildings that touch the outline of clusters, the orientation is also taken into consideration (see Appendix G.3 for the pseudo code).

- i. Follow the steps (i) to (iii) of the cluster shape algorithm I.
- ii. Get the orientation of each building and sort the orientation values which lie between 0 and $\pi/2$ according to the wall orientation algorithm by Duchêne *et al.* (2003) in ascending order.
- iii. Get the orientation difference ($d\theta$) between the maximum value and the minimum value.
- iv. If the $d\theta < 45^0$, go to step (ix).
- v. If the $d\theta > 45^0$, count the number of buildings with the orientation less than 45^0 (count_1) and that of greater than 45^0 (count_2). The value 45^0 is chosen because it is the reference orientation value used in deriving orientation between each topologically adjacent pair of buildings in the clustering algorithm explained in Section 5.2.1.
- v. If count_2 \geq count_1, iterate through each orientation less than 45^0 and add 90^0 so that all the orientation values are greater than 45^0 .

- vi. If count_1 > count_2, iterate through each orientation greater than 45^0 and subtract each from 90^0 so that all orientation values are less than 45^0 .
- vii. Now sort the modified orientation values in ascending order.
- viii. Subtracting the maximum value from the minimum value, Get new **dθ** which is considered as the **od** between buildings that touch the cluster outline.
- ix. If the **dθ** is within the threshold to be considered to have buildings oriented in the cluster, check orthogonality of each building in terms of the confidence interval (shape of the building outline) in the cluster outline using the algorithm by Duchêne *et al.* (2003). If the confidence interval of all the buildings that touch the cluster outline ≥ 80 , record that cluster as an orthogonal cluster in the spatial database, or else record that cluster as a non-orthogonal cluster.
- x. If the **dθ** is not within the threshold, record that cluster as non-orthogonal in the spatial database.
- xi. Iterate through the steps (i) to (x) until the end of all clusters in the query.

Evaluation of the results of the shape enrichment algorithm II

Figure 5.21 shows the results of the cluster shape enrichment algorithm II, which will be evaluated in comparison with the results obtained with the cluster shape enrichment algorithm I.

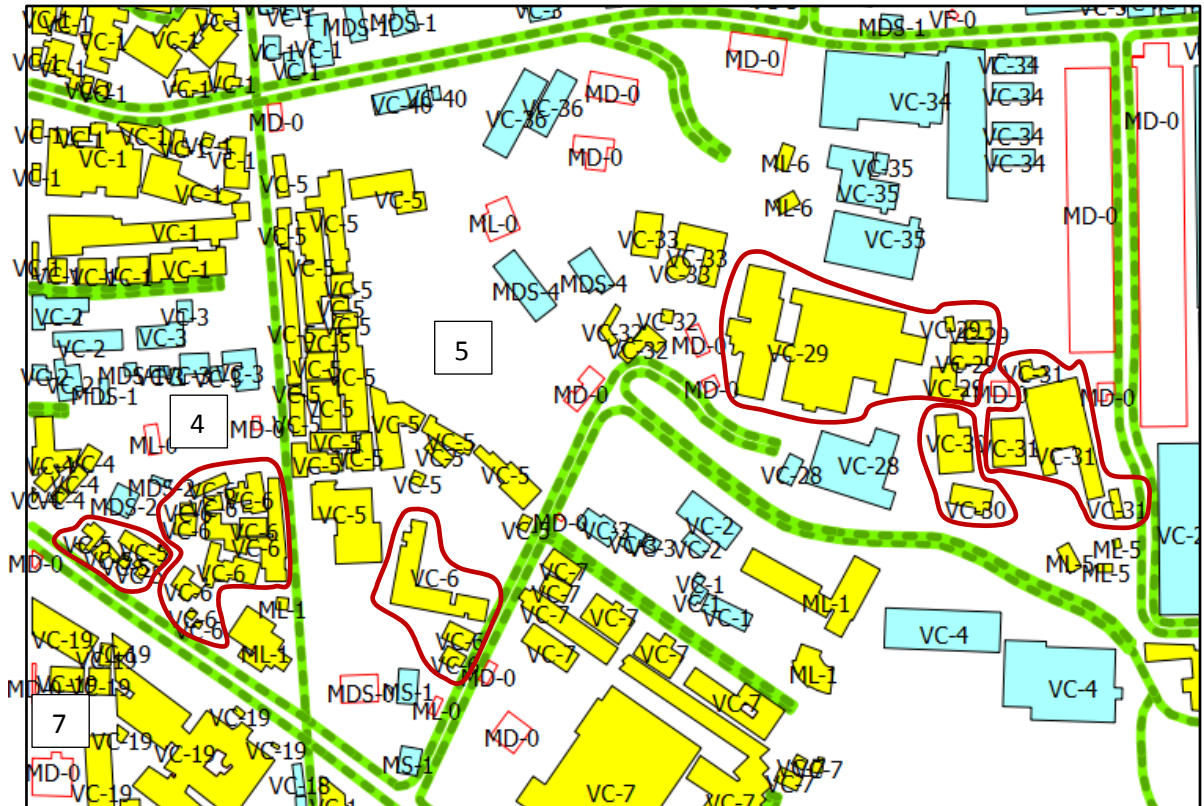


Figure 5.21 Shape enrichment of clusters in the same three regions 4, 5 and 7 (part of) surrounded by the road network as shown in Figure 5.20 with some improved results: clusters of orthogonal shape are shown in cyan colour while clusters of non-orthogonal shape are shown in yellow colour with the cluster label of each building shown in the cluster. Data source: 1 : 1K data of the NMA (Sri Lanka). Copyright reserved.

Table 5.15 Enlarged view of the clusters delineated in Figures 5.20 and 5.21 for the clear view of the orientation difference: clusters (a) and (c) are within region 4 and clusters (b), (d), (e) and (f) are within region 5.



The orientation threshold difference used in the prototype (see Figure B.7 in Appendix B.5) is 7° in the application of this algorithm, which is same as the value used in the application of hierarchical clustering for the data set. When comparing the shape enrichment results obtained using the shape enrichment algorithm I with that of obtained using the shape enrichment algorithm II, it is clearly identified that the clusters with labels 'VC-5' and 'VC-6' in region 4 and 'VC-6', 'VC-29', 'VC-30' and 'VC-31' in region 5 classified as orthogonal clusters (Figure 5.20) according to the algorithm I are reclassified as non-orthogonal clusters using the algorithm II according to the results in Figure 5.21. When observing the enlarged view of these clusters in Table 5.15, it is clear that the buildings that touch the cluster outline do not have a uniform orientation. Thus, the cluster shape enrichment algorithm II gives promising results of cluster shape characteristics, and it is the algorithm used in this research for cluster shape enrichment for the subsequent generalization process. The prototype developed with this algorithm enables one to apply different thresholds for orientation difference to derive different results in each region of the data set so that the different threshold values can be tested and used depending on the characteristics of data.

5.4.3 Synopsis of the contributing algorithms used

Table 5.16 summarises the contributing algorithms used in the main cluster shape enrichment algorithm.

Table 5.16 Cluster shape enrichment algorithm with the contributing algorithms: (1) new algorithm coupled with concave hull implementation with JTS (Section 5.4.1) and (2) wall statistical weighting algorithm by Duchêne *et al.* (2003).

Main algorithm	#	Contributing algorithms	Purpose
Cluster shape enrichment algorithm II	1	Improved concave hull based algorithm**	To select the buildings touching the cluster outline
	2	Wall statistical weighting*	<ul style="list-style-type: none">- Calculation of orientation of a building- To check the orthogonality of buildings

* Existing algorithm

** Modified and/or extension to an existing algorithm

5.5 Data enrichment workflow

Figure 5.22 depicts the sequence of operations for creating clusters and deriving cluster shape enrichment for subsequent automatic map generalization. The source data are held in the PostGIS database, and the results of both enrichment processes are written back to the spatial database for viewing through the front-end which in this research is the open source QGIS software.

After creating clusters to suit the target generalized map scale using the three Gestalt factors - proximity, orientation difference and similarity difference index for shape and size - users have the facility to re-run the process with different thresholds for the three Gestalt factors by viewing and testing the results until the results are satisfactory. Once the clustering process is completed, users can run the cluster shape enrichment with the input being the derived clusters stored in the spatial database. In this process, users have

the flexibility to view and test the results using different orientation threshold values in each region and write the results back to the spatial database.

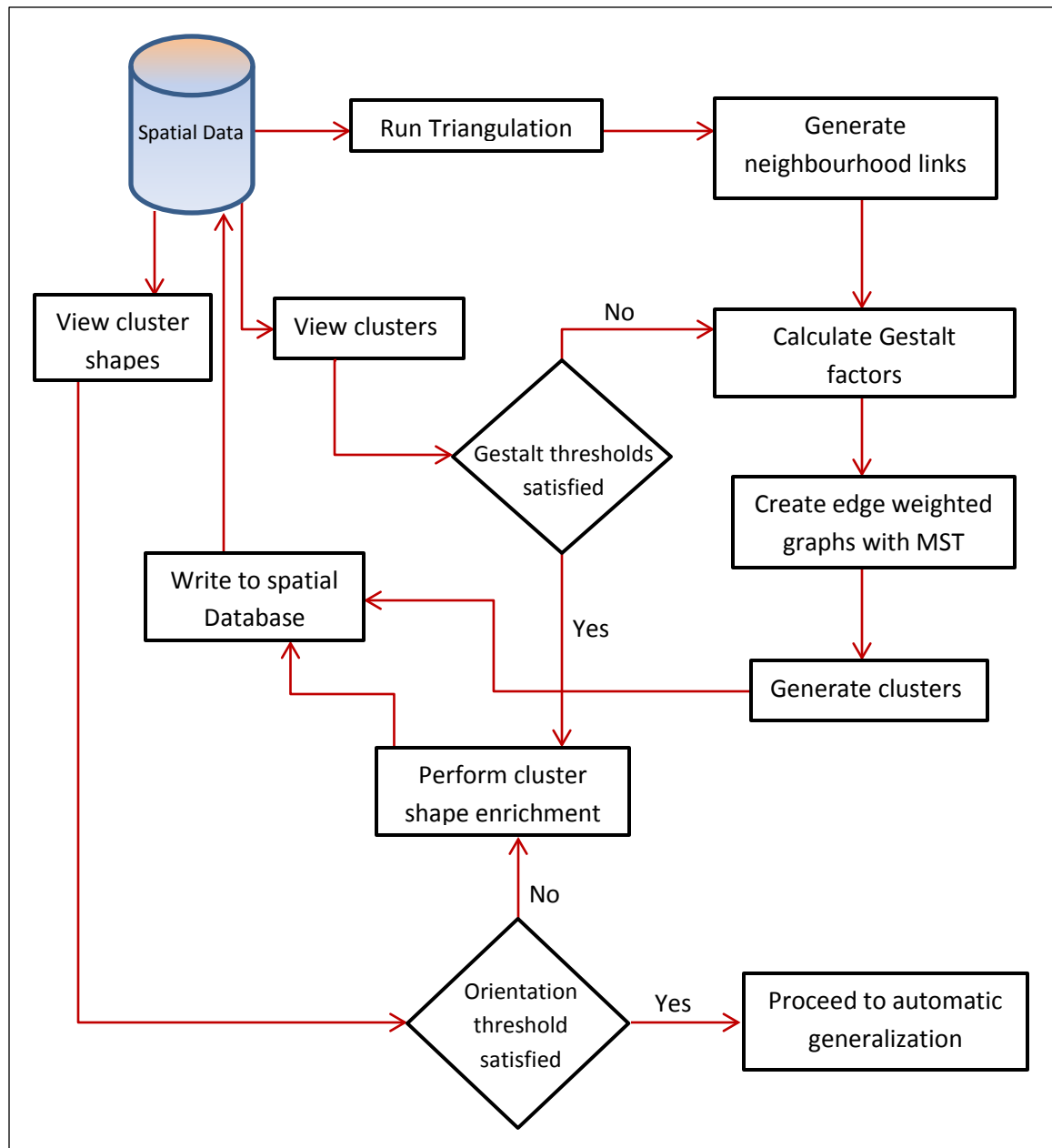


Figure 5.22 Data enrichment workflow of creating clusters and their subsequent shape enrichment.

5.6 Conclusion

This chapter presents and describes how the new algorithms are developed, tested and refined both in creating spatial clusters and deriving cluster shapes for subsequent automatic map generalization using existing algorithms with both synthetic and real building polygon data sets. The algorithm developed for spatial clustering has been validated with the two experiments conducted with the participation of subjects. The outcome of the analysis of the experiments reveals that the adaptation of manual hierarchical clustering is difficult due to varying perceptions of the subjects in distinguishing differences in orientation and similarity in shape, and in handling high cognitive loads in analysing the Gestalt factors. The majority of the subjects (about 87%) in both expert and lay groups have stated that the automatic hierarchical clustering is good. Also, all the subjects have realised the usefulness of the cluster classification employed in the automatic hierarchical clustering for subsequent map generalization. The intention of this experiment was to get an insight into the efficiency and the effectiveness of automatic hierarchical clustering in comparison with the manual hierarchical clustering. Thus, the subjects were not directly questioned if the automatic hierarchical clustering was better than the manual hierarchical clustering. However, the analysed results of their observations in both phases emphasise that the automatic hierarchical clustering brings much more prominent results than the manual hierarchical clustering. Further, the cluster shape enrichment results have been compared and evaluated qualitatively with the two algorithms developed in this research. When analysing, the cluster shape enrichment algorithm II gives promising results.

The next chapter will describe in detail the further implementation of data enrichment tools and their testing, together with the customisation of existing data mining algorithms for deriving landmark saliency. Deriving landmark saliency under data mining is a similar process to data enrichment to the topographic data (buildings and roads) used in this research and therefore the results of clustering and landmark saliency provide a strong support in the application of the generalization operators for subsequent automatic map generalization discussed in Chapter 7.

Chapter 6 Implementation - III: Data Mining Process

This chapter presents the data mining (knowledge discovery) process in generating salient landmarks to be depicted on focus maps. Knowledge of salient landmarks provides additional support for decision-making during the automatic map generalization process as to which features are to be retained and which are to be removed based on their saliency when features are represented at smaller scales. Data mining techniques by Sester (200b), Elias (2003) and Elias, Hampe and Sester (2005) have been used in the past for the knowledge discovery process of spatial data as discussed in Section 2.5.1. Research by Elias (2003) and Elias, Hampe and Sester (2005) used the ID3 classification algorithm and the COBWEB clustering algorithm to derive landmark saliency of building features. Statistical approaches using a combination of attributes have been adopted by Raubal and Winter (2002) and Nothegger and Winter and Raubal (2004) as discussed in Section 2.5.3. However, no comparison has been made between the two algorithms using real data sets as to select the best algorithm for deriving landmark saliency. As a result, the main focus of this chapter is to test, modify and compare three existing data mining algorithms - COBWEB, ID3 and C4.5 (C4.5 not previously tested in mining spatial landmarks) - with both synthetic and real data sets using open source WEKA data mining software (Weka, 2013) in order to derive saliency of buildings as prominent landmarks. The implementations have been customised by modifying the Java source code which is freely available in the WEKA data mining algorithm library with a new user interface (UI) (see Appendix F.4).

The inputs to these algorithms are the semantic, visual (geometric) and structural (spatial) characteristics of building features. In order to derive such characteristics from building features, data enrichment to be used in data mining is carried out as an initial phase. The second phase is the testing and comparison of the results (internal validation) of salient landmarks obtained from the three algorithms in the data mining process. The testing platform is the open source Java object oriented programming language with data stored in PostGIS, a spatial extension to PostgreSQL object-relational database management system to handle spatial data (see Appendix E.1 for SQL queries).



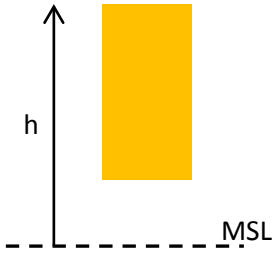


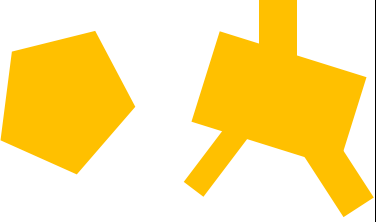
6.1 Data enrichment for the Data Mining process

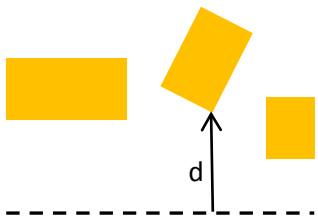
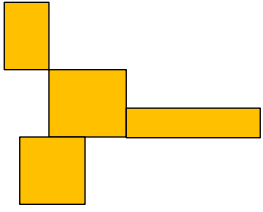
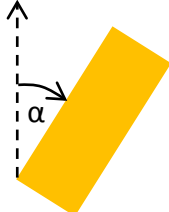

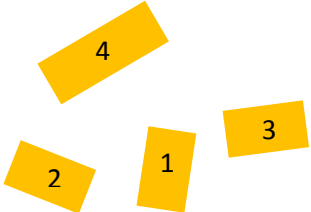
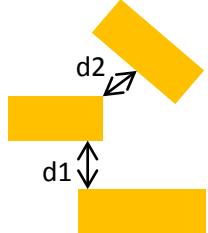
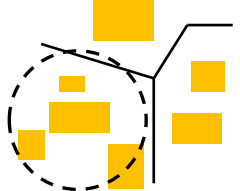
For the extraction of values of important attributes belonging to visual (geometric) and structural characteristics, functions are implemented in the same data enrichment prototype developed for data enrichment as discussed in the previous chapter and using most of the algorithms already discussed. For deriving structural properties, especially in relation to contextual features such as roads, the main algorithm used is the conforming Delaunay triangulation (CNDT) with the enforcement of edge constraints. The advantage of creating the CNDT over the DCT in dealing with buildings and roads is that in the CNDT, Steiner points are introduced at regular intervals on roads in the triangulation, and therefore it guarantees to capture the adjacency relationship with each and every building in relation to roads. If the DCT is to be used to find adjacency relationships of roads with buildings, roads have to be split into segments, introducing new vertices at a regular interval in order to make sure to capture all adjacency relationships between each and every building and roads as already done by Regnauld and Revell (2007) and Ai *et al.* (2007) in their applications in the field of automatic map generalization. For the extraction of semantics and visual properties - building size and height - existing attribute values stored in the building data set itself are used. Methods are developed to extract other attributes automatically from the spatial database as described and dealt with next.

6.1.1 Automatic derivation of attribute values

Brenner and Elias (2003) and Elias (2003) have suggested a set of attributes that are required to derive landmark saliency from a spatial database as discussed in Section 2.3.4. According to the availability of data, the attributes used in this research together with their description with pictorial representations are given in Table 6.1. Further, the methods and/or algorithms used for their value extraction are described in detail in this section.

Table 6.1 Description of attributes considered to derive salient landmarks in the data mining process.

#	Attribute property	Attribute	Value	Visual representation
i	Visual attraction	Corner	Number of corners	
ii		Size	Length (a) x Width (b) in [m ²]	
iii		Height	Elevation above the mean sea level (MSL) to the top of a building [m]	
iv		Elongation	Ratio between Width (b)/ Length (a)	
v		Orthogonality index	Edges orthogonal to each other - value = {1 (orthogonal), 0 (non - orthogonal)}	
vi		Diversely oriented edges	Edges with different orientations - value = {1 (diverse), 0 (non - diverse)}	

vii	Structural attraction	Distance to road	Minimum distance to road(s) [m]	
viii		Neighbours	Number of adjoining neighbours	
ix		Orientation	Orientation to North [degrees]	
x		Orientation to road	Parallel (longest edge along the road), across (width), angular to road, corner and none (orientation is irrelevant)	
xi		Orientation to neighbours	Average of orientation difference between neighbours [degrees] - for building 1: (difference [1,2] + [1,3] + [1,4])/3	
xii		Neighbour distance	Closest distance to adjacent buildings [m]	
xiii		Neighbourhood density	Density within close proximity [1/m ²]: Ratio between no. of buildings within a circular area around a building	
xiv	Semantic attraction	Importance	Hierarchy of priority of building use Value = {1, 2, 3, 4, 5}	

i. **Number of corners (both concave and convex)**

This is calculated by counting vertices from where the angle difference between succeeding line and preceding line $> 15^0$.

ii. **Size**

Area of each building is available in the source data set.

iii. **Height**

Height is considered to be the elevation from the MSL to the top of each building, obtained from the digital elevation model (DEM) in the data set.

iv. **Elongation**

Elongation is calculated by working out the ratio between the width and the length of the MBR of each building. These values range between 0 and 1 according to the equation. Higher the value lower is the elongation and vice versa.

v. **Orthogonality index**

This is calculated using the algorithm by Duchêne *et al.* (2003) described in Section 5.2.1 based on the wall (edge) orientation of a building. In this method, a building with orthogonality index $\geq 80\%$ is considered to be orthogonal and a building with the index value $< 80\%$ is considered to be non-orthogonal. Attribute value of orthogonality index (ϵ) assigned is: $\epsilon = \{1, 0\}$, where 1 denotes orthogonality and 0 denotes non-orthogonality.

vi. **Diversely Oriented edges (walls)**

This is also calculated using the same algorithm by Duchêne *et al.* (2003) using the weights of the candidate orientations. According to the algorithm, if a building has two leaves or more (a leave is the number of groups of spokes of different lengths, covering an angle of $\pi/2$ derived when calculating the statistical wall orientation of a building (Duchêne *et al.*, 2003)) and its orthogonality index is $< 80\%$, such a building is supposed to have edges

(walls) with several different orientations. An attribute value of diversely oriented edges (ϵ) is assigned a binary number, $\epsilon = \{1 \text{ or } 0\}$, where 1 denotes diverse orientation, and 0 denotes the non-diverse orientation of the edges.

vii. Minimum distance between a building and roads

- a. Iterate through the adjacency list of buildings and roads obtained from the CNDT with constraint edges and get the first iteration.
- b. Get the closest distance between a building and a road which would form a pair and add each building in the pair into a map containing building IDN and the closest distance in a list. In this operation, if the building IDN is already available in the map, add the closest distance only against existing building IDN in the list.
- c. Go to the next iteration in step (a) until the end of the list.
- d. Now using the building IDN required to find the closest distance to the roads, retrieve the corresponding list containing all the closest distances between the building and roads from the map.
- e. Sort the list in ascending order and get the first element of the sorted list as the closest distance to the roads from the corresponding building.

viii. Finding adjoining neighbours (adjoining buildings)

- a. Iterate through the adjacency list between buildings obtained from the CNDT with constraint edges and get the first iteration.
- b. Find if a building has zero distance to the other building. If yes, add each building in the pair into a map containing building IDN and the distance in a list. In this operation, if the building IDN is already available in the map, add the distance only against existing building IDN to the list. If the distance between the buildings in a pair is not equal to zero, such links are not added.
- c. Go to the next iteration in step (a) until the end of the list.

- d. Using the building IDN required, retrieve the corresponding list from the map and find out the number of zero distances attached against each building IDN.
- e. Calculate the number of zero distances (number of 0 distances are equal to the number of adjoining neighbours). Buildings that are not attached to the list are considered to have no adjoining neighbours.

ix. **Orientation to North**

This attribute helps identify buildings which have different orientations from a group of buildings of similar orientation. It is considered as the orientation of the edge (wall) of the building with the maximum statistical weight according to the algorithm by Duchêne *et al.* (2003).

x. **Orientation to road**

A new improved algorithm has been developed to determine the orientation of a building to the road in this research, adopting the algorithm I given below, followed by its further refinement.

Algorithm I

- a. Iterate through the adjacency list containing [bidg_id, road_id, min_dst] of each building to roads.
- b. If a particular building has three or more neighbouring roads in the adjacency list, such a building is considered to be at a corner as illustrated in Figure 6.1(a).

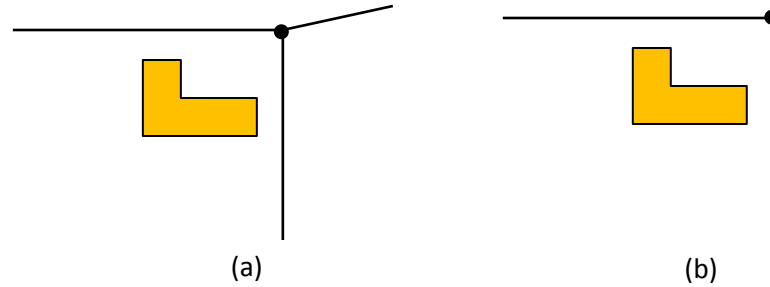


Figure 6.1 Building at a corner: (a) three road segments meet at a corner, and (b) two road segments meet at a corner.

- c. If a building has two neighbouring roads that intersect each other, such a building is considered to be at a corner (Figure 6.1(b)).
- d. If a non-corner building (building IDNs 1 or 2 in Figure 6.2) has two neighbouring road segments on either side which do not intersect each other, select the road with the minimum distance to the building out of the two roads and adopt the following algorithm.

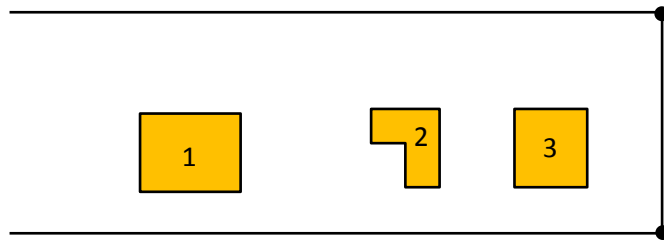


Figure 6.2 Buildings with three neighbouring road segments around (Buildings with IDNs 1 and 2: non-corner buildings and building with IDN3: a corner building).

1. Get the wall orientation using the algorithm by Duchêne *et al.* (2003).
2. Rotate the building using the wall orientation to orient it along the X-Y axes.
3. Calculate the MBB of the rotated building and rotate it back to the same orientation to get the locally oriented bounding rectangle (LOBR) of the building. This is the MBR oriented in the direction of the wall orientation.

4. Compute a distance factor (DF) by taking into consideration the minimum distance between the building and the road and a constant distance called the visual perception distance (currently taken as 50m). If the minimum distance is less than or equal to 50 m, assign the $DF = 1$. If it is greater than 50m, assign the $DF = 50m$.
5. Calculate four outward segments parallel to the longest side of the LOBR from its four corners with the distance of each segment equal to the distance obtained by multiplying the minimum distance from the DF.
6. Check if one of the two outward pairs of line segments from the two opposite longest sides intersects the line segments of the road near the building by iterating the line segments of the road through.
7. If it does not intersect, get one of the two outward pairs drawn parallel from the two opposite shortest sides, which intersects the road (Figure 6.3(a)) and adapt the following steps:
 - Iterate through each line segment of the road and get the angle subtended by both intersecting outward segments.
 - If both angles subtended are outside $90^\circ \pm 15^\circ$ between each outward segment and road line segment, the building is considered to be angular (Figure 6.3(b)).
 - If both angles subtended is within $90^\circ \pm 15^\circ$ between each outward segment and road line segment, the building is considered to be parallel to that particular segment (Figure 6.3(a)).
 - If one angle is within $90^\circ \pm 15^\circ$ and the other is outside, the building is considered to be angular-parallel to a road as illustrated in Figure 6.3(c).
8. If the pair of outward-line segments parallel to the longest side does intersect a road, iterate through the road line segments and get the two

angles with the similar threshold values applied in step 7 above. As in the case of step 7, if both angles are within $90^\circ \pm 15^\circ$, the building is considered to be across the road (not physically, but virtually, see Figure 6.3 (d)). If both angles are outside $90^\circ \pm 15^\circ$, the building is considered to be angular (Figure 6.3(e)) and If one angle is within $90^\circ \pm 15^\circ$ and the other is outside, the building is considered to be angular-across to the road as illustrated in Figure 6.3(f).

d. Go to the next iteration in step (a).

The algorithm uses a constant distance, considering the visual perception distance (taken as 50m) to extend the outward line segments towards the road. The intersection of these segments depends on the visual perception distance.

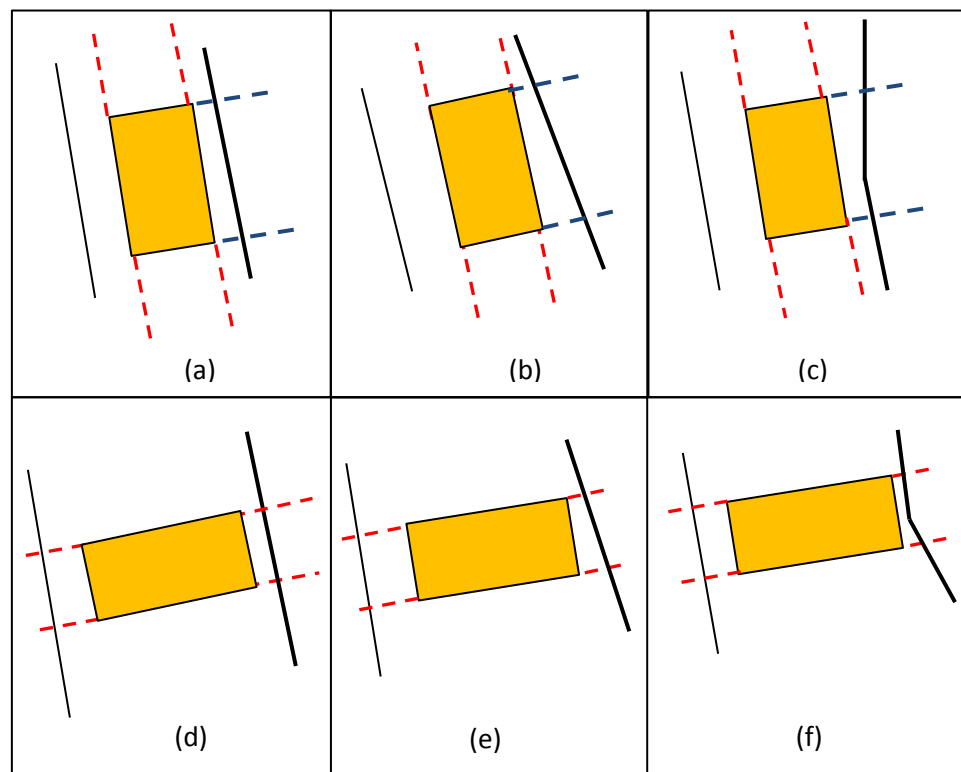


Figure 6.3 Different orientation of the LOBR of a building to the closest road: (a) parallel (b) angular (c) angular-parallel (d) across (e) angular and (f) angular-across.

Improved algorithm

In this approach, the algorithm I is improved for determining the intersection of outward segments of the LOBR with road segments independent of visual perception distance by extending each segment of the LOBR iteratively to meet the road.

- a. Adopt steps (a) and (b) from algorithm I.
- b. If a particular building has two neighbouring roads (Figure 6.2) which do not intersect each other, select the road with the minimum distance to the building out of the two roads and adapt the following algorithm.
 1. Adapt sub-steps I, 2 and 3 in step (d) in the algorithm I.
 2. Create two line segments connecting the centroid to the midpoint of the two perpendicular edges of the LOBR (Figure 6.4).

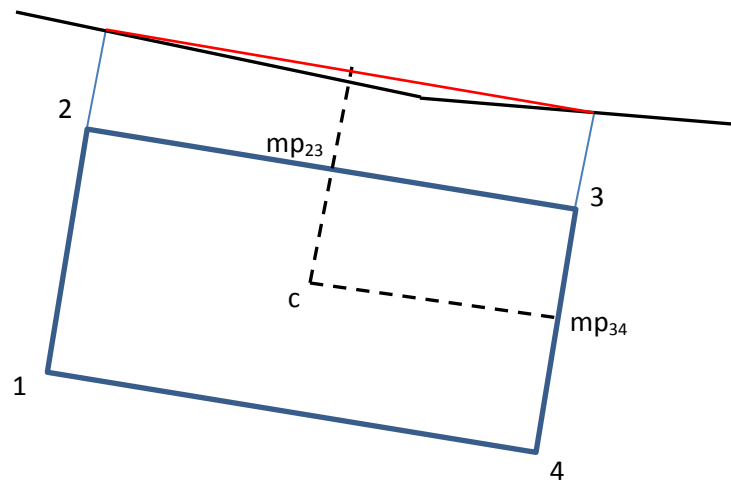


Figure 6.4 LOBR of the building with two line segments connecting the centroid and midpoints of its edges.

3. Check if these two segments intersect with the road by extending them iteratively outward from both ends.

4. If both line segments intersect the road, find the length of each extended segment from the centroid to the intersection point on the road and go to the next step, or else go to step 8.
5. If the length of the extended segment from the centroid perpendicular to the longest edge of the LOBR (edge 23 in Figure 6.4) is smaller than the length of the other extended line segment from the centroid perpendicular to the shortest edge, adapt the following steps:
 - Extend the two edges (edges 12 and 43 in Figure 6.4) perpendicular to the longest edge and get the two intersecting points on the road.
 - Get the angle difference between the line joining two intersecting points (red line in Figure 6.4) and the extended line segment (extended segment cmp_{23} in Figure 6.4).
 - If the angle difference is within $90^0 \pm 15^0$, the building is considered to be parallel to the road.
 - If the angle difference is outside $90^0 \pm 15^0$, the building is considered to be angular to the road.
6. If the length of the extended segment perpendicular to the longest edge of the LOBR (edge 23 in Figure 6.4) is greater than the length of the other extended line segment, adapt the first two sub-steps under step 5, extending two edges perpendicular to the shortest edge of the LOBR.
 - If the angle difference is within $90^0 \pm 15^0$, the building is treated to be across the road virtually.
 - If the angle difference is outside $90^0 \pm 15^0$, the building is treated to be angular to the road.
7. Go to step (a).

8. If the line segment from the centroid, extended perpendicular to the longest edge of the LOBR only intersects the road, extend the two edges of the LOBR perpendicular to the longest edge, find the angle difference between the line connecting two intersection points on the road and the extended line segment from the centroid and adapt the following steps:
 - If the angle difference is within $90^0 \pm 15^0$, the building is considered to be parallel to the road.
 - If the angle difference is outside $90^0 \pm 15^0$, the building is considered to be angular to the road.
9. If the line segment extended perpendicular to the shortest edge of the LOBR only intersects the road, extend the two edges of the LOBR perpendicular to the shortest edge, find the angle difference between the line connecting two intersection points on the road and the extended line segment from the centroid and adapt the following steps:
 - If the angle difference is within $90^0 \pm 15^0$, the building is considered to be across the road virtually.
 - If the angle difference is outside $90^0 \pm 15^0$, the building is considered to be angular to the road.
10. Go to step (a).

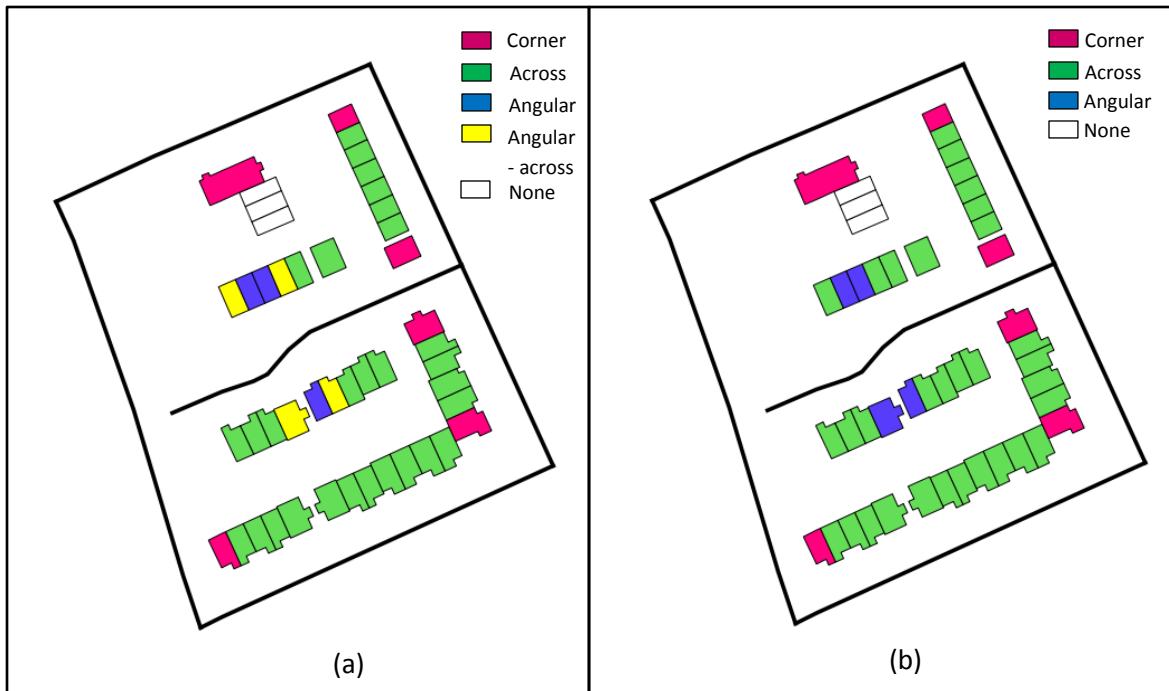


Figure 6.5 Results of building orientation to the road on the OS MasterMap Data within a region surrounded by the roads: (a) results of the algorithm I and (b) results of the improved algorithm.

Evaluation of the results of the orientation to the road

According to Figure 6.5, the orientation to the road of the three buildings is 'none' as they do not spatially connect to roads in the triangulation. This categorisation of orientation in the algorithm I was too descriptive (seven categories - corner, parallel, across, angular, angular-parallel, angular-across and none), hence revised in the improved algorithm. In the improved algorithm, only five categories (corner, across, parallel, angular and none) are considered. Figure 6.5 depicts some of the categories of orientation to the road derived from the two algorithms.

xi. Orientation to neighbours

- Iterate through the adjacency list between buildings obtained from the CNDT with constraint edges and get the first iteration.
- Get the building orientation difference between the two buildings in a pair and add each building in the pair into a map containing the building IDN and the

orientation difference stored in a list. In this operation, if the building IDN is already available in the map, add only the orientation difference against existing building IDN to the list.

- c. Go to the next iteration in step (a) until the end of the adjacency list.
- d. Retrieve the corresponding list from the map and find out how many orientation differences are attached in the list against the building IDN from which it is required to find the average orientation difference to neighbours. This number reflects the number of adjacent neighbours to the building.
- e. Calculate the average orientation difference of the corresponding building, dividing the total orientation difference by the number of adjacent neighbours.

xii. **Neighbour distance**

- a. Iterate through the adjacency list between buildings obtained from the CNDT with constraint edges and get the first iteration.
- b. Get the closest distance between the two buildings in the pair and add each building into a map containing building IDN and closest distance in a list. In this operation, if the building IDN is already available in the map, add only the closest distance against existing building IDN in the list.
- c. Go to the next iteration in step (a) until the end of the list.
- d. Using the building IDN required to find the closest distance to neighbours, retrieve the corresponding list containing all the closest distances between each pair of buildings from the map.
- e. Sort the list in ascending order and get the first element of the sorted list as the closest distance to other building neighbours.

xiii. **Neighbourhood density**

This is considered as the ratio between number of buildings divided by the area of the region around these buildings as defined by the radial distance (currently taken as 50m, but can be changed depending on the nature of the data) of a circle with its centre being the centroid of the building of which density is to be calculated.

This is achieved using an update query (see Appendix E.1(4)) in PostGIS using its **ST_DWithin** function. The **ST_Centroid** function is used to get the centroid of a building. The query itself updates the attribute field called “neigh_density” in the building layer.

xiv. **Building importance**

A priority value based on the building function/use is assigned to each building. For this purpose, the priority type of each building given in the building source data, is categorised into distinctive groups and then a priority is assigned to buildings in each group as given in Table 6.2.

Table 6.2 Building categories and their priority rankings.

#	Category	Priority
1	Attraction (Cultural and Historical, Botanical and Zoological, Recreational, Tourism, Pubs and Retail shops)	1
2	Health	2
3	Educational	
4	Public Infrastructure	
5	Transport	
6	Sports and Entertainment	
7	Commercial	3
8	Manufacturing	4
9	Residential	5

6.2 Data mining approach

The three data mining algorithms - COBWEB hierarchical cluster, ID3 and C4.5 (known as the J48 implementation in the WEKA data mining software) decision trees - are tested and evaluated. Three important pre-processing steps need to be carried out on the enriched attributes before applying algorithms for mining salient building landmarks; the first step being to handle missing attribute values, the second step is the data transformation and the third step being the sensitivity analysis. These three processes are described next.

6.2.1 Data pre-processing

Handling missing values

Depending on the type of the decision tree used for data mining, missing values, if found in the data set, have to be treated beforehand. The customised ID3 decision tree algorithm code available in the WEKA data mining software used in this research does not handle missing values. One of the methods of handling a missing value is to treat it as another value of the attribute if the missing value is significant when analysing data. However, if there is no particular significance of the attribute value, a more appropriate solution is to ignore (remove) all instances in which some of the values are missing (Witten, Frank and Hall, 2011). This option is not advisable in some cases where the instances with missing values often provide a good deal of information because of the significance of other attribute values. Another option is to work out the mean or probable value, considering other values. Handling of missing values in this research will be discussed under the testing phase of the ID3 decision tree in the WEKA data mining software.

Attribute data transformation

Depending on the algorithm chosen for data mining, different attribute domains (types) are required. In the application of the ID3 algorithm as discussed in Section 2.5.1, it cannot handle numeric values. Therefore, the transformation of numeric attributes to nominal and/or ordinal values is necessary. For example, building size can be assigned either small or large (ordinal value) instead of a numeric value. In some occasions, even if the algorithm can handle numeric values, there are instances where more effective results can be obtained by classifying the numeric values into nominal. In the WEKA data mining software, automatic attribute transformation methods are available for this process and will be discussed in the testing phase of the ID3 algorithm below.

Sensitivity analysis

It is necessary to identify which attributes make a significant impact on the decision tree output before applying a decision tree algorithm. The reason is that there can be one or more attributes that make no significant contribution to the output (e.g. if the attribute values are the same for all the objects). The WEKA data mining software has the facility to check the suitability of the selected attributes to be considered for classification with an 'Attribute selection' menu with full automation. It gives a list of ranks in order of significance of the attributes. This will be discussed in the testing phase of the three algorithms.

6.2.2 Testing of the data mining algorithms on a synthetic data set

The three algorithms - COBWEB, ID3 and J48 - already implemented in the WEKA GUI are tested on a synthetic data set of a decision point given in Figure 6.6 to understand how salient buildings are represented in the output produced by the WEKA software. In this example in Figure 6.6, it is clear that building 3 (building IDN - BP3) with the highlighted outline is a salient feature.

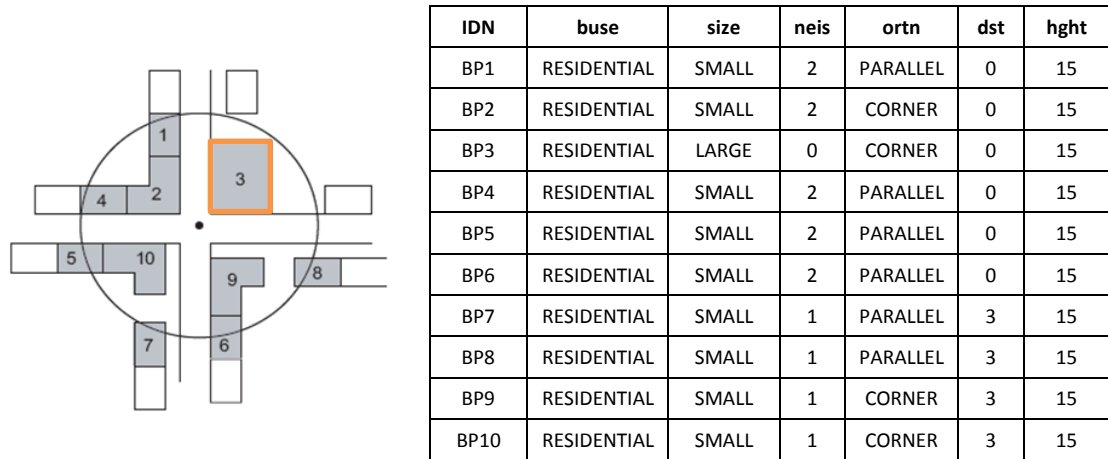


Figure 6.6 Synthetic data set with its attributes at a decision point, based on Elias (2003): buse - building use, neis - number of adjacent neighbours, ortn - orientation of a building in relation to road, dst - minimum distance to road and hght - building height.

COBWEB hierarchical clustering

The COBWEB clusterer is a hierarchical clustering and an unsupervised learning method as described in Section 2.5.1. Hence, no explicit training examples are needed. The COBWEB algorithm implemented in the WEKA software can handle both numeric and nominal data. Hence, the data transformation is not always necessary. It can also deal with missing values.

In the synthetic data set given in Figure 6.6, no missing values are available to be dealt with. However, sensitivity analysis is required to check the significance of the attributes in the final results. Hence, the automatic sensitivity analysis is carried out using the filtered attribute evaluation function (see Appendix F.1 for sensitivity analysis workflow used in the WEKA GUI). When investigating the ranked attributes (see Appendix F.2), it can be seen that the attribute fields - 'Building use' and 'Height' - are ranked the lowest. The reason is that the attribute values in these two attributes are common to all and do not carry any discerning values. Therefore, these two attributes are ignored before applying the COBWEB clusterer in the WEKA software.

Application of the COBWEB clustering

The COBWEB clustering is applied to the synthetic data set given in Figure 6.6 after removing the two attributes during sensitivity analysis. The synthetic data set is used as the training data set (the data set used to train the classifier) with the class attribute - 'IDN' - used for cluster evaluation. The necessity of using this clusterer is not to group building objects into clusters, but to identify salient landmarks through the tree structure and the cluster information as discussed by Elias (2003).

Analysis of clustering on the synthetic data set

When investigating the cluster output given in Figure 6.7, it is possible to identify the cluster IDN of each instance clustered (see classes to cluster classification matrix in Figure 6.7(right)) and the number of instances in each cluster (see clustered instances in Figure 6.7(left)). Further, it is understood that the graphical cluster output of the tree view given by the WEKA software is not informative. With the tree, it is not possible to identify which attributes are assigned to nodes with attribute values used to split the tree and what objects belong to which tree level. From the graphical representation of the tree in Figure 6.8, what is only identified is the number of instances assigned to each node and a further count on how they are split into leaves in the tree. However, the analysis of the cluster output and the matrix shown in Figure 6.7 together with attributes in Figure 6.6 enables one to manually enrich the graphical tree view with the details of attributes, their values and the instances, a node or a leaf holds on the tree (see Figure 6.8). The node and leaf numbers given in the graphical output represent the cluster IDN when analysing with the cluster classification matrix which represents the cluster IDN against the class IDN (instance IDN).

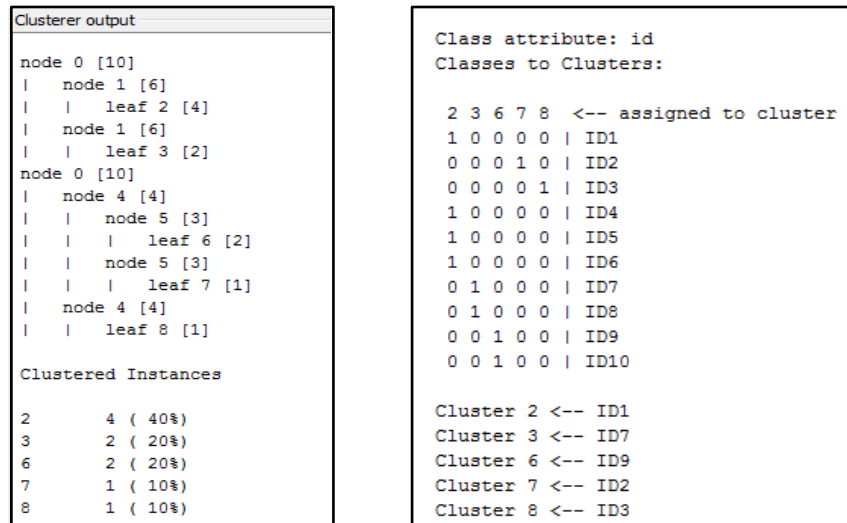


Figure 6.7 COBWEB clustering tree view (left) and the matrix showing cluster numbers against instances (right) of the synthetic data set in Figure 6.6.

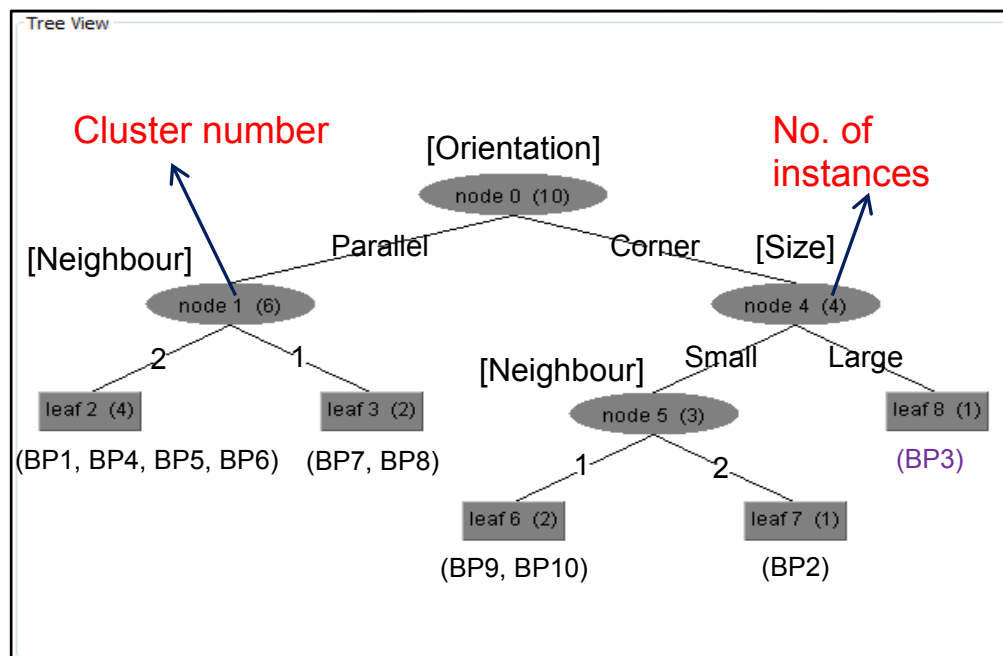


Figure 6.8 Graphical representation of an enriched tree view with attributes, their values and instance IDNs (class IDNs) after the analysis.

When analysing the tree further, it is found that the salient landmarks get isolated towards the top level of the tree (at a higher level containing a single instance in a cluster, e.g. building with class IDN BP3 in Figure 6.8). It is also clearly visible that the algorithm

aggregates instances with similar attribute values into clusters. The more deviations in the attribute values, the greater is the depth of the tree.

ID3 decision tree

It is a supervised classification algorithm (supervised learner) that learns from the training data to make predictions for new data. It can only deal with nominal attributes. No missing values are allowed. The aim of the ID3 algorithm is to create leaf nodes with homogeneous data based on the information gain of individual attributes as described in Section 2.5.1.

Application of ID3

The synthetic data set does not have missing values and therefore it is not necessary to deal with missing values. Since ID3 cannot handle numeric data, numeric attributes in the data set need to be transformed into a smaller number of distinct ranges. This is also called attribute discretization according to Witten, Frank and Hall (2011). However, depending on the discretization method, classification results vary. Unsupervised discretization is used in the absence of any knowledge of the classes of instances in the training data set while supervised discretization takes the classes into account. In this research, the attribute values of landmark class are the values that are not known and to be derived. As such, unsupervised discretization is the best option to be adopted.

In the unsupervised discretization, there are two binning methods available in the WEKA software - equal-width binning and equal frequency binning. Equal width binning is sensitive to outliers in the data where it categorises neighbourhood values - 0, 1, and 2 - into 3 bins when applied (Figure 6.9(a)). Equal frequency binning assigns more or less the same number of training samples into each bin, adding more smoothness to the data (equal frequency binning classifies neighbourhood values - 0 and 1 - into a single class as shown in Figure 6.9(b)). This is also called the histogram equalisation due to the reason that when a histogram of the contents of the resulting bins is investigated, it gets completely flat. However, this method greatly affects the ability of the attribute to help build good decision trees, identifying the discerning attributes in the application of

deriving salient landmarks in this research. As such equal-width binning under unsupervised attribute discretization is used to transform numeric attributes to nominal attributes in applying the ID3 decision tree on the test data (see Appendix F.3).



Figure 6.9 Transformation (discretization) of the attribute - neighbour - in the synthetic data set in the WEKA GUI: (a) discretization with equal-width binning and (b) discretization with equal frequency binning.

The next step is the sensitivity analysis of the data set. The two attributes - 'Building use' and 'Height' - are ignored from the data set used for classification based on the sensitivity analysis as already applied to the COBWEB clustering algorithm.

The ID3 decision tree is a supervised classification algorithm, and therefore it is required to have known classes in the data set. In the test data set, the landmark attribute is used as the class attribute. Since its attribute values are not known, it is iteratively hypothesised each building to be a landmark, starting from the first instance in the data set during the classification process (see Table 6.3).

Table 6.3 Instances with attributes chosen after the sensitivity analysis on the synthetic data set where the attribute **lmark** is the classifier used as a dependent variable. The first instance is initially assigned to be a landmark which is iteratively assigned to each instance down the tuples during classification. Note: Attributes - **neigh** and **dist** - should be discretized to be nominal values before applying the ID3 classification.

IDN	size	neigh	orient	dist	lmark
BP1	SMALL	2	PARALLEL	0	YES
BP2	SMALL	2	CORNER	0	NO
BP3	LARGE	0	CORNER	0	NO
BP4	SMALL	2	PARALLEL	0	NO
BP5	SMALL	2	PARALLEL	0	NO
BP6	SMALL	2	PARALLEL	0	NO
BP7	SMALL	1	PARALLEL	3	NO
BP8	SMALL	1	PARALLEL	3	NO
BP9	SMALL	1	CORNER	3	NO
BP10	SMALL	1	CORNER	3	NO

NO
YES
↓

Analysis of the classification on the synthetic data set

When the classifier is run hypothesising each instance to be a landmark in a manual iterative process, the instance with IDN BP3 on the data set is correctly classified as a landmark (Figure 6.10(b)). It lies at the topmost level in the tree. None of the other instances are classified as a landmark (see Figure 6.10(a) for the classifier output at the first iteration). However, the WEKA software does not generate a graphical view of the ID3 implementation.

<pre> Classifier output === Classifier model (full training set) === Id3 neigh = '(-inf-0.5]': NO neigh = '(0.5-1.5]': NO neigh = '(1.5-inf)': NO orient = PARALLEL: NO orient = CORNER: NO Time taken to build model: 0 seconds === Evaluation on training set === === Summary === Correctly Classified Instances 9 90 % Incorrectly Classified Instances 1 10 % Kappa statistic 0 Mean absolute error 0.15 Root mean squared error 0.2739 Relative absolute error 64.2857 % Root relative squared error 89.1133 % Total Number of Instances 10 </pre>	<pre> Classifier output Test mode: evaluate on training data === Classifier model (full training set) === Id3 size = SMALL: NO size = LARGE: YES Time taken to build model: 0 seconds === Evaluation on training set === === Summary === Correctly Classified Instances 10 100 % Incorrectly Classified Instances 0 0 % Kappa statistic 1 Mean absolute error 0 Root mean squared error 0 Relative absolute error 0 % Root relative squared error 0 % Total Number of Instances 10 </pre>
---	--

(a)

(b)

Figure 6.10 Classification output of the synthetic data set: (a) output at first iteration with the first instance hypothesized to be a landmark and (b) output at third iteration with the third instance hypothesized to be a landmark.

The classification output also gives a statistical error analysis of each instance. From the analysis of the tree, it can be deduced that a landmark is characterised by a short decision tree with only a few levels that lead to a potential landmark decision-making.

J48 decision tree

The J48 implementation in the WEKA software is also a supervised classification method based on the C4.5 algorithm by Quinlan (1993), which is a further improvement to the ID3 algorithm as discussed in Section 2.5.1. Not only it can deal with missing values, but also with both numeric and nominal attributes and has a more robust splitting of attributes via gain ratio, including pruning of the tree structure. However, in this research, the unpruned tree of the J48 implementation for the classification of landmarks is used since each instance needs to be tested.

Application of the J48 implementation

The first two steps that are essential to the ID3 algorithm to handle missing values and transform numeric attributes into nominal values are not required in this algorithm. Since sensitivity analysis has already been done on the synthetic data set, attributes that are to be removed are already known ('Building use' and 'Height'). Thus, the classification can be directly tested. However, hypothesising each building to be a landmark as described under the ID3 classification (Table 6.3) is applied to the J48 implementation since it is also a supervised classification method.

Analysis of the classification on the synthetic data set

When the classifier is run hypothesising each instance to be a landmark in a manual iterative process as in the classification with the ID3 algorithm, the instance with IDN BP3 is correctly classified as a landmark (Figure 6.11). Further, it lies on one of the leaves at the topmost level in the tree. None of other instances are classified as a landmark. Similar to ID3, with the J48 implementation, a landmark is characterised by a short decision tree with only a few levels for potential landmark decision-making.

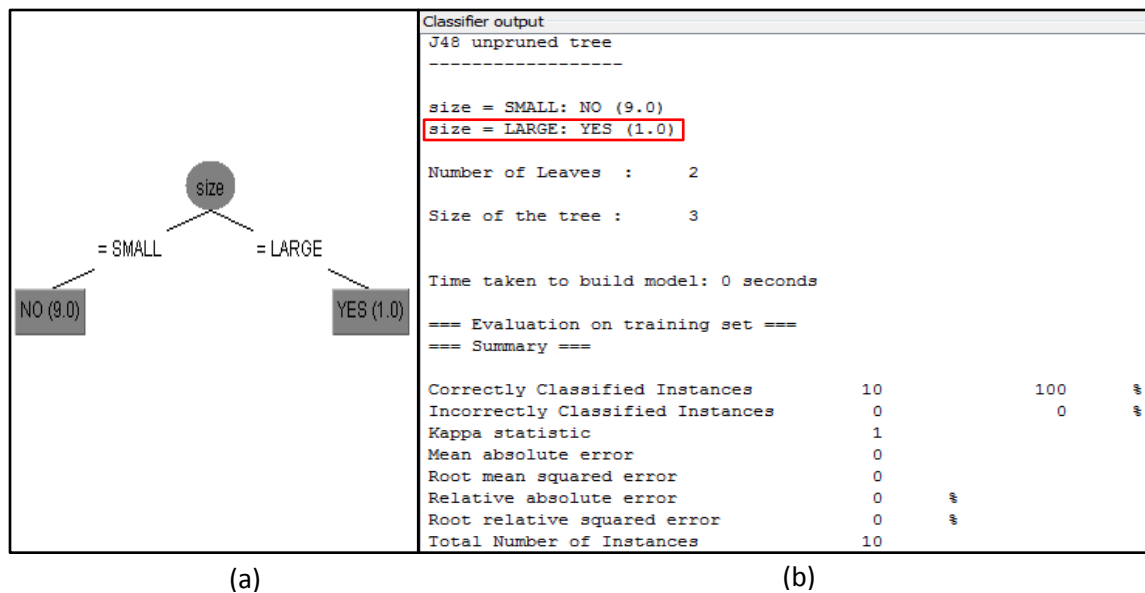


Figure 6.11 Classification output of the synthetic data set at the third iteration with the third instance hypothesized to be a landmark: (a) graphical representation of the tree view and (b) the tree information.

Implications of the analysis of the three algorithms

The three algorithms - ID3, COBWEB and C4.5 with J48 implementation - tested on the synthetic data set at the decision point, all identify building IDN BP3 as the salient landmark at the topmost level of each tree of the three algorithms. It is the design of each tree of these three algorithms that the top of the tree is the most important landmark (this is the performance measurement of the three algorithms). In the case of the ID3 and the J48 implementations, salient landmarks are always on the top of a short decision tree (Figures 6.10(b) and 6.11(a)). In the case of COBWEB clustering, salient features get isolated towards the top of the tree (building IDN BP3 in Figure 6.8). The same characteristics of the trees of ID3 and COBWEB algorithms in retrieving salient landmarks have been emphasised by Elias (2003). The WEKA software gives similar results in applying the three algorithms when observing the final result of landmark saliency (building IDN BP3). However, the WEKA software does not have functions to traverse through each tree of ID3, COBWEB and J48 implementations and pick up the objects identified as salient features from the top level of the tree. Also, the iterative assignment of attribute values in the dependent variable of the instances (class attributes **Imark** in this example) of the instances is not facilitated by the WEKA software.

6.2.3 Testing of the data mining algorithms on a real data set

Considering the limitations found in the WEKA software to capture salient landmarks from the outputs of the three algorithms - COBWEB, ID3 and C4.5 with J48 implementation, the source code of these algorithms are extended with new methods to traverse through the output tree and pick the salient landmarks needed with a new UI (see Appendix F.4) based on the performance measurement of each tree as discussed under implications of the analysis of these algorithms in the previous section. The results in deriving salient landmarks are visualised to verify the impact of the performance measurement adopted. The developed interface that has the facilities to read data from a spatial database or a file with one of the different formats (.csv, .arff etc.) can deal with attribute transformation, ignoring of some attributes (removal of insignificant attributes) and writing salient landmark information back into the spatial database. However, the sensitivity analysis functions are not implemented in this UI. They are used from the WEKA software.

Data set

For testing the three algorithms with the new UI, a data set is considered within a region surrounded by the roads (data set in Figure 6.5, see part of the data set in ASCII format in Appendix F.5) from the OS MasterMap data at the scale of 1 : 1.25K which is enriched with the attributes (listed in Table 6.1) derived using the implementation of data enrichment methods discussed under Section 6.1.

Missing values are generated in spatial data sets in finding the minimum distance to roads from buildings and orientation to the road from buildings in cases where buildings are not spatially connected to the road network in the triangulation as mentioned under the evaluation of results of building orientation to the road in Section 6.1.1. In this research, a single dummy value (-1) is assigned to all the attribute values with NULL in the attribute field - 'mindist_road' (minimum distance to the road) - and a dummy value 'none' to the attribute field - 'orientation_road' (orientation to the road) - to treat them as another possible unique value at the time of data enrichment under Section 6.1. This is a suitable approach rather than devising alternative methods to avoid null values either by ignoring

instances comprising of missing values or working out the most probable value or mean as discussed in Section 6.2.1. The reason is that there is a possibility where such instances can be candidates to become salient landmarks with other attributes. Further, applying a statistical value would not be realistic, especially with attributes comprising of spatial characteristics. Treating null values is especially required in this testing phase because the ID3 decision tree algorithm used in this research to emphasise landmark saliency cannot handle missing values. However, in order to keep consistency, treating null values is carried out, before executing all the three data mining algorithms during testing.

Having treated NULL values as explained above, sensitivity analysis is carried out as applied to the synthetic data set (see Section 6.2.2), and five attributes - 'diverse_sides' (sides with different orientations), 'orthogonal' (orthogonality of sides), 'neigh_density' (neighbourhood density), 'av_ortn_neigh' (average orientation to neighbours) and 'importance' (priority on building use/function) - are ignored as a result. The three algorithms are tested thoroughly with four explicit versions of the data set: (a) original data set with no discretization (except ID3 algorithm) (b) original data with discretization (c) data with rounded off decimal values for the three attributes - size, the minimum distance to road, and elevation - with no discretization (except ID3 algorithm) and (d) rounding off the same attribute data with discretization. Testing the algorithms with rounding off values of the three aforesaid attributes is necessary since there are several close values when investigating the data set. Each algorithm tested, and the results obtained by traversing the tree from the top to the bottom on the four versions of the above data set are given next.

Analysis of the results of the COBWEB clustering

Discretization of rounded off data has made a minor impact in the results when observing Table 6.4 in the application of COBWEB clustering on the top level of the tree. When observing the results of COBWEB using the combination of the top two levels (Table 6.5), attribute discretization has made a significant improvement both in the original data set and the data with rounded off decimal values of the three attributes. However, rounding of values of these three attributes in the data set has produced poor results when

compared with the results (a) and (b) with that of (c) and (d) in Table 6.5. This implies that the COBWEB clusterer is sensitive to the precision of attribute values.

Table 6.4 Results of the COBWEB clustering on the top level of the tree: (a) original data with no attribute discretization (b) original data with discretization of the three attributes - size, the minimum distance to road, and elevation (c) data with rounded off values of the three attributes with no discretization and (d) data with rounded off values of the three attributes with discretization. Salient landmarks are shown in red colour.

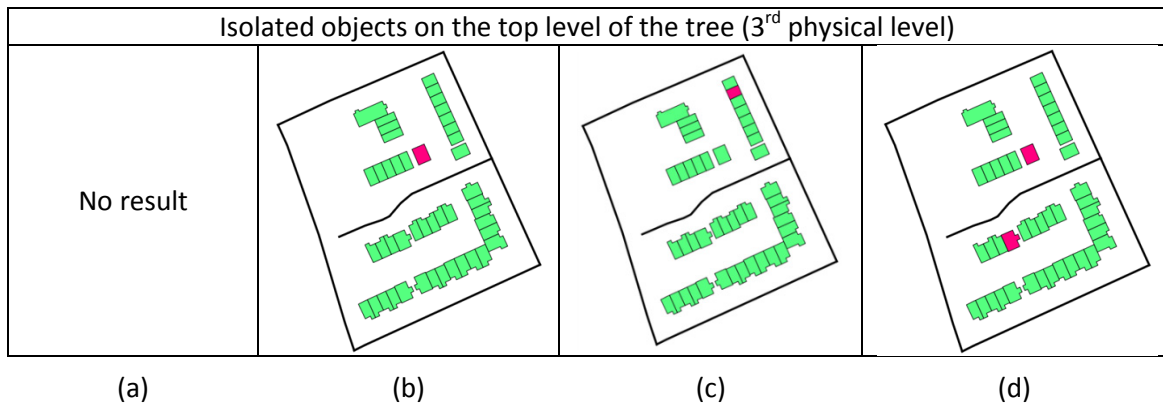
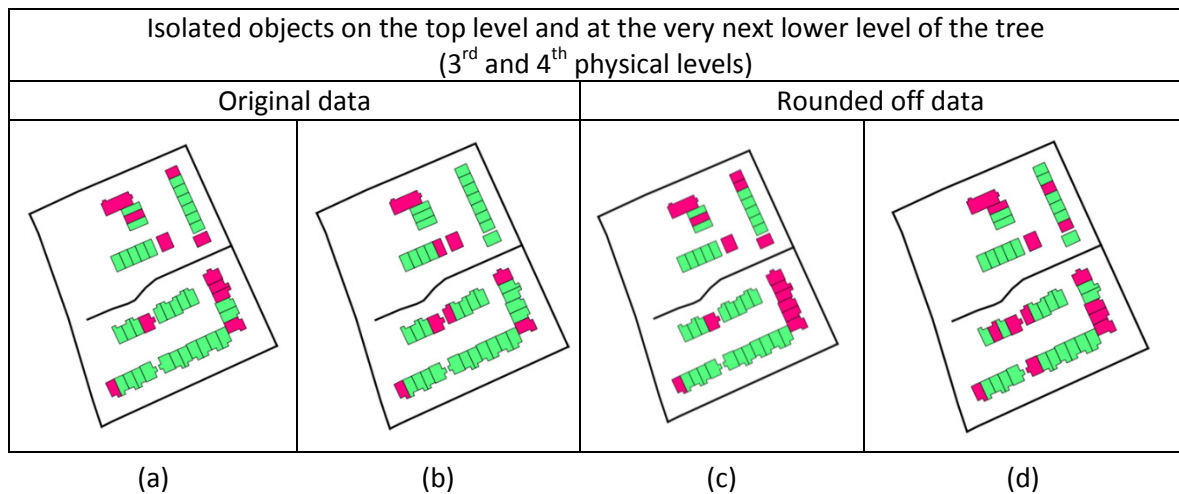


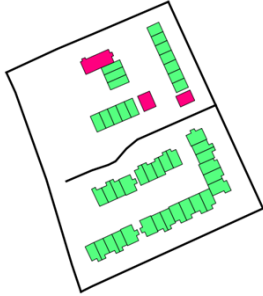
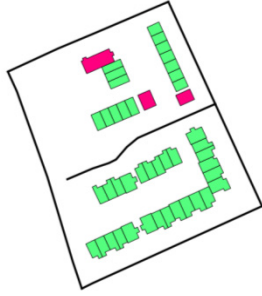
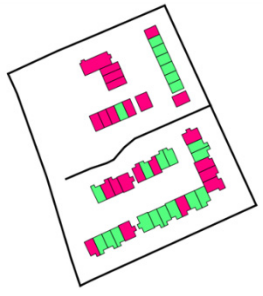
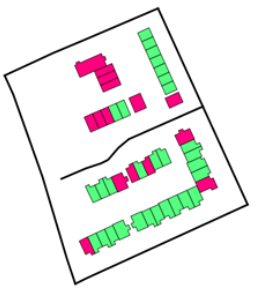
Table 6.5 Results of the COBWEB clustering using the top two levels of the tree: (a) original data with no attribute discretization (b) original data with discretization of the three aforesaid attributes (c) data with rounded off decimal values of the three aforesaid attributes with no discretization and (d) data with rounded off decimal values of the three aforesaid attributes with discretization. Salient landmarks are shown in red colour.



Analysis of the results of the ID3 classification

There is no significant difference in results between the original data and the rounded off data on the top level of the tree (Table 6.6(a) and 6.6(b)). However, rounded off data have produced rather improved results when the top two levels are combined when comparing results of Table 6.6(c) and 6.6(d).

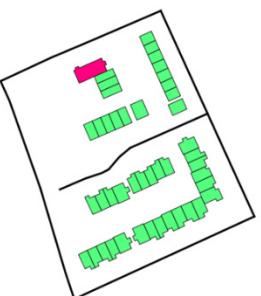
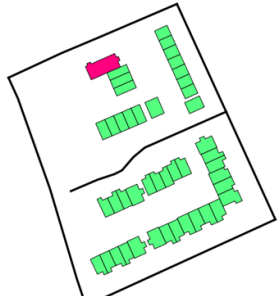
Table 6.6 Results of the ID3 classification: (a) original data with a discretization of all numeric attributes (b) data with rounded off values of the three aforesaid attributes and discretization of all numeric attributes on the top level of the tree (c) original data with a discretization of all numeric attributes and (d) data with rounded off decimal values of the three aforesaid attributes on the top two levels of the tree. Salient landmarks are shown in red colour.

Isolated objects on the top level of the tree (1 st physical level)		Isolated objects on the top level and at the very next lower level of the tree (1 st and 2 nd physical levels)	
Original data	Rounded off data	Original data	Rounded off data
			
(a)	(b)	(c)	(d)

Analysis of the results of the J48 classification

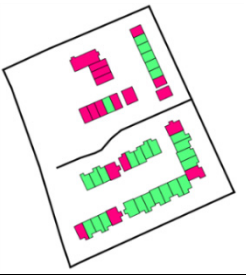
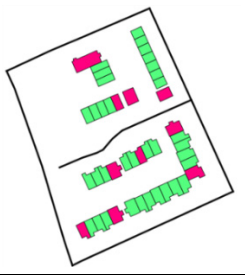
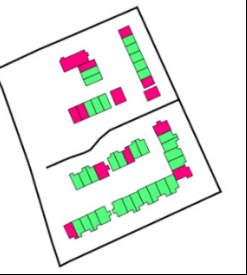
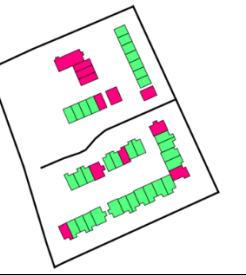
The original and the rounded off data without discretization have not produced results on the top level of the tree while both data sets with discretization have produced the same results (Table 6.7).

Table 6.7 Results of the J48 classification on the top level of the tree: (a) original data with no attribute discretization (b) original data with discretization of the three aforesaid attributes (c) data with rounded off decimal values of the three aforesaid attributes with no discretization and (d) data with rounded off decimal values of three aforesaid attributes with discretization. Salient landmarks are shown in red colour.

Isolated objects on the top level of the tree (1st physical level)			
Original data		Rounded off data	
No result		No result	
(a)	(b)	(c)	(d)

The J48 implementation has produced improved results using the combination of the top two levels with attribute discretization of the original data (Table 6.8(b)) when comparing the results obtained with attribute discretization in Table 6.7. The rounded off data without attribute discretization have also given improved results (Table 6.8(c)) when comparing with the results of data without attribute discretization (Table 6.8(a)). However, rounding off of data with attribute discretization has produced coarse results, selecting four adjoining buildings (see Table 6.8(d)).

Table 6.8 Results of the J48 classification using the top two levels of the tree: (a) original data with no attribute discretization (b) original data with discretization of the three aforesaid attributes (c) data with rounded off decimal values of the three aforesaid attributes with no discretization and (d) data with rounded off decimal values of the three aforesaid attributes with discretization. Salient landmarks are shown in red colour.

Isolated objects on the top level and at the very next lower level of the tree (1 st and 2 nd physical levels)			
Original data		Rounded off data	
			
(a)	(b)	(c)	(d)

Implications of the analysis of the three algorithms

When comparing the visual results of Table 6.4, 6.6 and 6.7, it is observed that the performance measurement (i.e. the top level of the tree is the salient landmark) used in the synthetic data set as a decision point has not worked out to derive salient landmarks in the application of COBWEB, ID3 and J48 implementations when these have been applied to data representing a region. In such instances, the performance measurement that has to be applied is the combination of levels down the tree, starting from the top to the required level. Tables 6.5, 6.6(c) and 6.6(d), and Table 6.8 show the generated results using the combination of the top level and its very next lower level of the tree in the real data set. However, the number of levels that have to be searched down the tree for salient objects (e.g. in COBWEB, an isolated object in a cluster, and in the ID3 and the J48 implementations, a classified object with **lmark** = 'yes') vary, depending on the data set.

6.2.4 Evaluation of the results of the three data mining algorithms

With the synthetic data set

When analysing the results of three algorithms on the synthetic data set in Section 6.2.2, the three algorithms are suitable to derive landmark saliency at decision points since the three algorithms have given a unique result which is building 3 (building IDN BP3) on the top level of each tree. This is the building with its outline highlighted, identified to be a salient landmark initially in the synthetic data set described in Section 6.2.2. The building with IDN BP3 is chosen because of its discerning characteristic - 'large' - in the attribute name - 'size'. With the ID3 and the J48 implementations, in addition to identifying salient landmark objects, characteristics leading to make a particular object as a salient landmark can be extracted from the tree. However, with the COBWEB clusterer only the salient landmark objects can be identified.

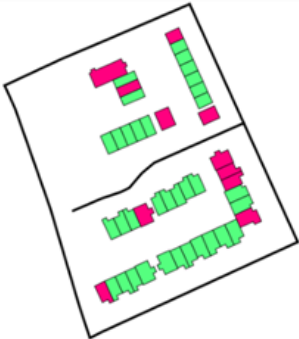
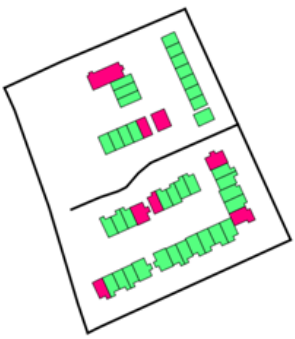
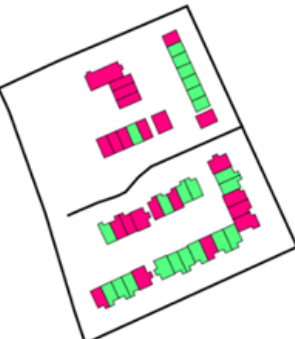
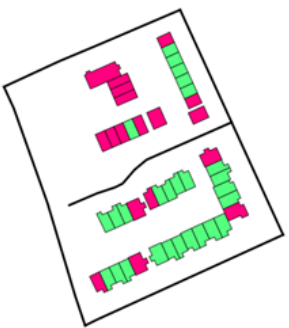
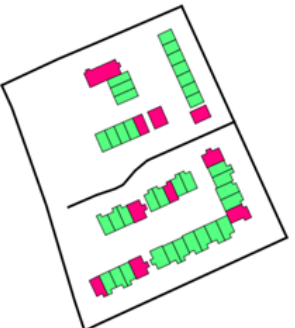
With the real data set

When investigating the results in Section 6.2.3, it is realised that the rounding of the attribute values has made no significant improvement in the results in the three data mining algorithms tested. However, the attribute discretization with equal-width binning has improved the results of deriving salient objects. Therefore, the comparison of results given in the previous Section 6.2.3 is made only with the original data with and without attribute discretization given in Table 6.9.

When comparing the results with the application of the three algorithms on the real data set given in Table 6.9 with buildings spreading over a region rather than at a decision point, it is understood that the ID3 algorithm is not suitable to apply to data spreading over a region or a larger area since it tends to select buildings that are not candidates to be salient landmarks.

The results of COBWEB and the J48 implementation of the C4.5 algorithm without attribute discretization reveal that the building objects comprising of attributes with close values tend to be chosen as landmarks when observing Table 6.9. One of the examples is the chosen four adjoining buildings which have elevations 12.36m, 12.9m, 12.59m and 12.44m in the results of the J48 implementation in Table 6.9.

Table 6.9 Comparison of the results of the three data mining algorithms using the top level and the very next lower level (1st and 2nd physical levels) of the output trees on a building data set spread over a region enclosed by roads. Salient landmarks are shown in red colour.

Algorithm	Visual representation of the results	
	Without discretization	With discretization
COBWEB		
ID3	-	
J48		

Comparing the results of the COBWEB and the J48 implementations, attribute discretization has had a significant effect on the results. The results of the COBWEB and the J48 implementations on discretization are more or less similar. However, the J48 implementation has the ability to identify more discerning attributes as potential landmarks (e.g. detached building at the corner where the three road segments meet) when observing the results of the COBWEB and the J48 implementations with attribute discretization.

Since the J48 implementation has given promising results of deriving salient landmarks both at a decision point and in a region with the ability to find over which attribute values an object becomes a landmark, the J48 implementation is chosen as the algorithm to derive salient landmarks in the generation of focus maps in this research.

6.3 Conclusion

This chapter initially describes how data are enriched for the subsequent use for deriving salient landmarks under data mining, developing new methods and algorithms including testing of results, taking into account the context of building objects such as roads and hydrographic features. In the second phase, the research tests three existing data mining algorithms - COBWEB hierarchical clusterer, ID3 and the J48 decision tree implementations - with further customisation for deriving salient landmarks from building features in the enriched data stored in the spatial database. The ID3 algorithm used by Elias (2003) at a decision point is not a successful candidate for deriving salient landmarks of a data set representing a region as investigated in this work. Further, the performance measurement of the trees adopted by Elias (2003) has not turned out promising results in deriving salient landmarks with COBWEB, ID3 and J48 data mining methods on a data set representing a region. It is identified that searching of objects in a combination of adjacent levels from the top level to the required level down the tree is required for deriving salient landmarks when applying these three algorithms on a data set representing a region. When testing the results, it can be concluded that out of the three algorithms, the J48 implementation gives promising results for deriving landmark saliency both at a decision point and in a region. These salient landmarks, thus generated, will be portrayed on a coarse background of building features in generating focus maps at the end of the thesis. The next chapter will describe in detail the implementation of tools for building data generalization to be applied in the enriched spatial clusters created with the use of the data enrichment process described in Chapter 5 for generating coarse background on the focus map. Information of the derived salient landmarks discussed in this chapter can also be used in the decision-making process for selecting building features to be depicted at the generalized scales.

Chapter 7 Implementation - IV: Automatic Map Generalization

Process

This chapter presents automatic map generalization processes required for deriving focus maps for wayfinding with the use of building geometries. The focus map generation will be realised by draping salient building landmarks on a generalized background through the application of the polygon aggregation operations on building geometries with the map generalization as discussed in Section 2.2.8. In this work, four types of polygon clusters are considered for the development of four different aggregation operations based on the cluster characteristics: (a) clusters which are smaller than a threshold area when aggregated are represented by symbols; (b) clusters with their outline almost orthogonal are aggregated with squared sides; (c) clusters with their outline non-orthogonal are aggregated by bridging the gaps between buildings in the cluster using a distance threshold, and (d) extensive clusters forming built-up areas are aggregated using a concave hull approximation. In addition to aggregation algorithms, a building simplification algorithm and a building enlargement algorithm, to be applied after aggregation, are also developed apart from the use of some existing algorithms for simplification. Creating novel algorithms and modifying existing algorithms is carried out with thorough testing and refinement based on the enriched building geometries processed using the data enrichment tools developed under the data enrichment process discussed in the previous chapter. In developing generalization algorithms, internal validation will be carried out with both synthetic and real data sets.

The testing platform is open source Java object oriented programming language with data stored in PostGIS, which is a spatial extension to PostgreSQL object-relational database management system to handle spatial data (see Appendix E.1 for SQL queries). Implementation of all the aforesaid aggregation algorithms is realised through a prototype development (see Appendix B.6).

7.1 Symbolization algorithm with squaring or enlargement

The use of this algorithm is to represent building clusters which are smaller than a certain distance threshold when approximated to a group oriented bounding rectangle (GOBR) as explained in Figure 7.1, depending on the target generalized scale with either a polygon square or rectangular symbol, maintaining the orientation of the source cluster at the target scale (see Appendix G.4 for the pseudo code).

- i. Create concave hull using the improved concave hull based algorithm explained in Section 5.4.1 on a building cluster.
- ii. Find the edges of buildings at the cluster outline that touch the concave hull by a line (e.g. buildings with IDNs 3, 4, 6, 7 and 8 touched by the concave hull as depicted in Figure 5.18(c)) in Section 5.4.1.
- iii. With the use of these touching edges of buildings in the cluster, calculate the maximum wall (edge) orientation of the cluster using the algorithm by Duchêne *et al.* (2003).
- iv. Rotate the cluster about its centroid to align it with its maximum wall orientation along the X-axis (Figure 7.1(a)).

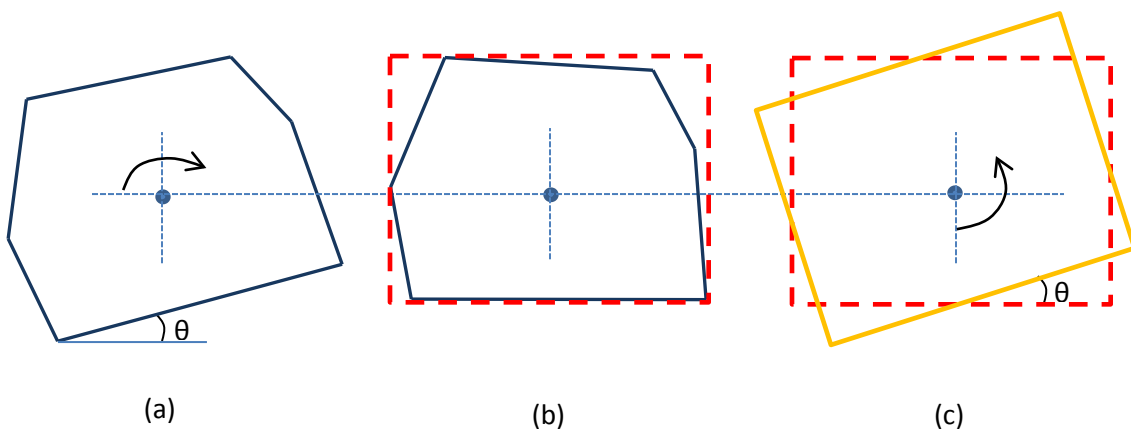


Figure 7.1 Generating group oriented bounding rectangle (GOBR): (a) outline of the cluster (b) MBB created after rotating cluster by the maximum wall statistical orientation (θ) and (c) GOBR created by rotating the MBB back with the same orientation in the opposite direction.

- v. Calculate the MBB of the rotated cluster of buildings. This is the MBB_{x-y} aligned along the X-Y axes, which closely fits with the cluster based on the maximum wall orientation (Figure 7.1(b)).
- vi. If both sides of the MBB_{x-y} are less than the minimum side length to represent at the target scale, the cluster is replaced by a square of the minimum side length, oriented according to the building group orientation by rotating about the centroid of the cluster back with the use of the maximum weighted orientation.
- vii. If only one side of the MBB_{x-y} is less than the minimum side length according to the target map scale, the shorter side is extended at both ends of the MBB_{x-y} to have the minimum side length and reform the rectangular geometry oriented to the X-Y axes (Figure 7.2).

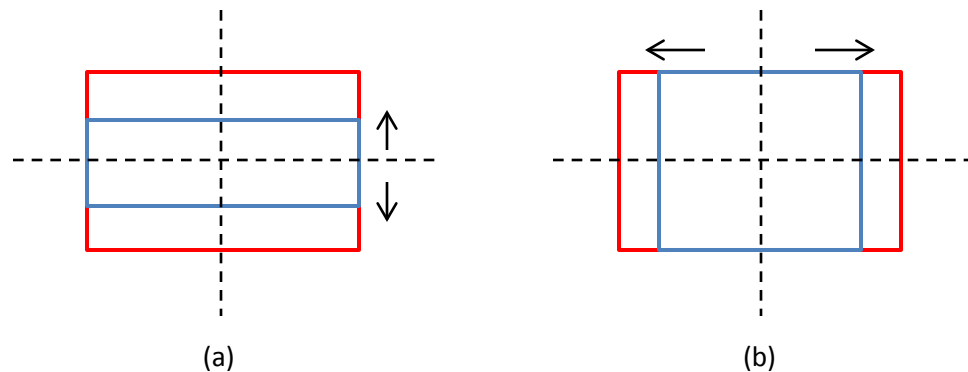


Figure 7.2 Building enlargement: (a) enlargement along the Y-axis (height) and (b) enlargement along the X-axis (width) to comply building edges with the minimum building length.

- viii. Finally, rotate rectangular geometry back about the centroid of the cluster with the use of the maximum weighted orientation to form the GOBR (Figure 7.1(c)).

7.2 Building cluster aggregation with orthogonal sides

The intention is to get a square amalgam by aggregating buildings in an orthogonal shaped cluster which is identified and enriched during the cluster shape enrichment process described in Section 5.4 under the data enrichment process. The complete realisation of the algorithm for building aggregation with orthogonal sides should undergo a sequence of applications of the four generalization algorithms: (a) filling the gaps to create the amalgam (b) squaring the amalgam (c) enlargement of narrow sections and juts, and (d) simplification of the granular edges to suit the target generalization scale. During the amalgam creation process, if the cluster is too small to be shown on the target scale when aggregated, it is symbolized in the target scale as described in Section 7.1. Testing of the aforesaid generalization algorithms, leading to the final evaluation of the main algorithm developed in this research together with its internal evaluation, is dealt with in the next sections.

7.2.1 Creating amalgam with dilation and erosion

This is based on the expansion and the shrinkage technique adopted by Schylberg (1992) on raster data. For expansion and shrinkage, the buffer operation on vector data is used.

Dilation and erosion with buffer operation I

- i. Read the union of each building cluster from the spatial database.
- ii. Dilate the union geometry with a distance slightly greater than the minimum threshold distance used for building clustering in the data enrichment process. Ensure to square the cap of the buffering geometry with the end cap style being CAP_SQUARE (3) and the join style of sharing buffer strips with JOIN_MITRE (2) to form flat edge corners and preserve right angle edges (Figure 7.3) in setting buffer parameters in the JTS library. These settings are required to preserve the orthogonality of the sides of the aggregated geometry influenced by the orthogonal sides of the buildings that touch the outline of a cluster that is subject to the aggregation process.

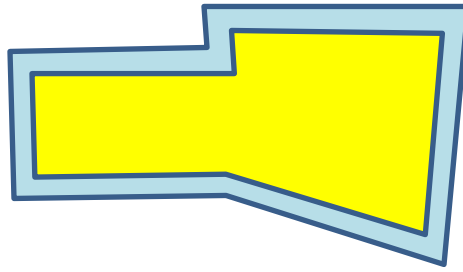


Figure 7.3 Example illustration of a buffer around a polygon with cap square and join style mitre.

- iii. Apply erosion with the same distance (negative buffer) and the parameters used in the dilation process to get the buffered amalgam of the buildings in the cluster. If the buffered amalgam is not a single polygon entity, especially in exceptional cases where the building geometries in a cluster have either overhanging, corner touching or total separation configuration (Figure 7.4), the dilation and erosion process with the same distance may create a split amalgam with multi-polygons.

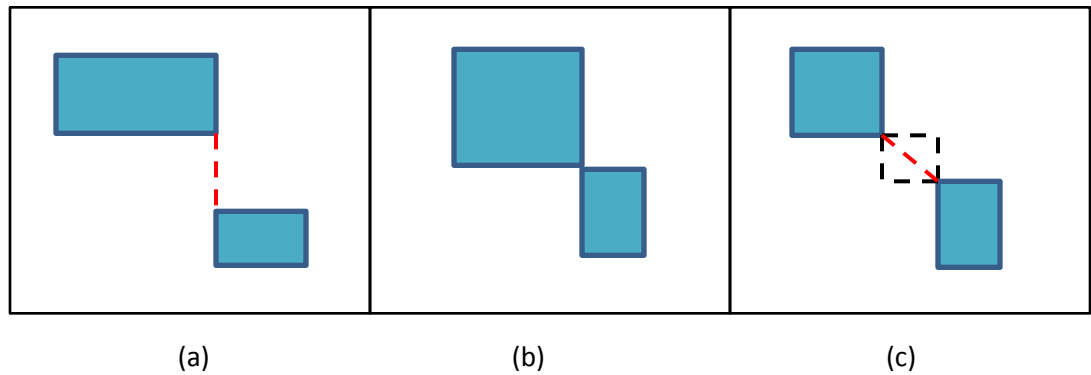


Figure 7.4 A pair of buildings in a cluster at different positions: (a) almost overhanging, (b) corner touching and (c) total separation.

Dilation and erosion with buffer operation II

- i. Follow the steps (i) to (iii) under the buffer operation I above.
- ii. If the buffered amalgam is not a single polygon entity, the erosion distance used in step (iii) in the buffer operation I is not assigned the same negative buffer distance as used in dilation in step (ii) in the buffer operation I. Instead, an eroding distance is applied iteratively by a small increment until the operation stops when it reaches the maximum possible eroding distance to form the buffered amalgam with a single polygon entity (Figure 7.5).
- iii. The distance difference left to apply for the total erosion is recorded for further use on the results of the final aggregation after enlargement before the simplification operation.

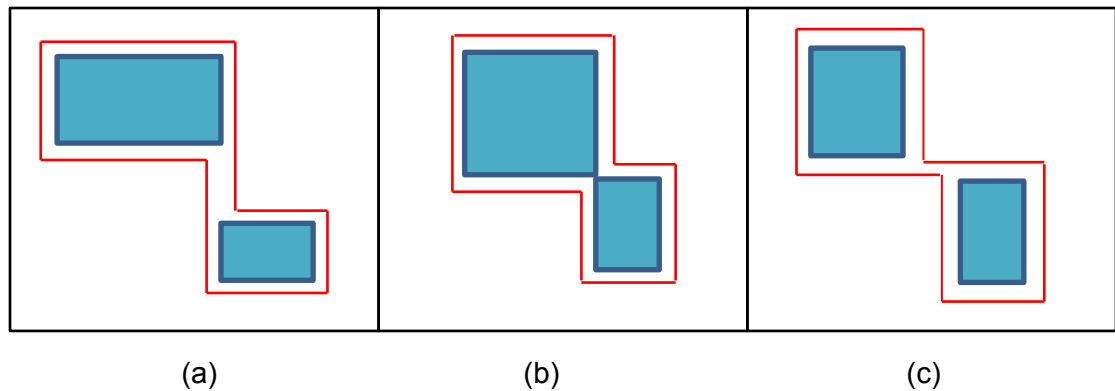


Figure 7.5 Polygon geometry shown in red colour using the dilation operation and then the erosion with iterative shrinking for exceptional cases where multi-polygons are created in blue colour with the same dilation and erosion distance: (a) almost overhanging (b) corner touching and (c) total separation.

It should also be mentioned that if a polygon that is subject to buffering has holes and the buffering distance applied is greater than the minimum distance between some holes and the outer ring, such holes are removed in the buffering process.

7.2.2 Creating amalgam with concave hull generation

- i. Read each building cluster from the PostGIS spatial database iteratively.
- ii. Create concave hull using the **ST_ConcaveHull** function with the target percent being the value of 0.99 based on the of area of convex hull, usually faster than forming a convex hull according to PostGIS 2.0 manual (PostGIS Manual, 2012). In generating the concave hull, the target percent can be a value between 0 and 0.99 (see Appendix E.1(6)). Generating the concave hull with a value of 1 gives its convex hull. The percent value of 0.99 is used because when lower values are used, generalization of the concave hull gets higher, thus creating an amalgam not suitable to generate a subsequent squared amalgam of building clusters.

7.2.3 Squaring edges of the amalgam

After applying either the buffering operation or the concave hull generation on each building cluster to create the initial amalgam, its outline may not be orthogonal due to orientation differences of the buildings that touch the outline of the cluster (Figure 7.6(a)). In order to make these edges orthogonal, the following squaring algorithms are developed and tested.

Squaring algorithm I

- i. Create the MBR of the clustered amalgam.
- ii. Calculate the orientations of the longest and the shortest sides of the MBR.
- iii. Iterate through each edge of the buffered amalgam and calculate both triangular polygons (ears) in and out of the clustered amalgam with the edge being common baseline to both triangles (Figure 7.6(a)) using the two orientations perpendicular to each other in the MBR. If either of the orientations of the MBR is equal to that of an edge of the amalgam, creating triangular polygons for that particular edge is ignored.

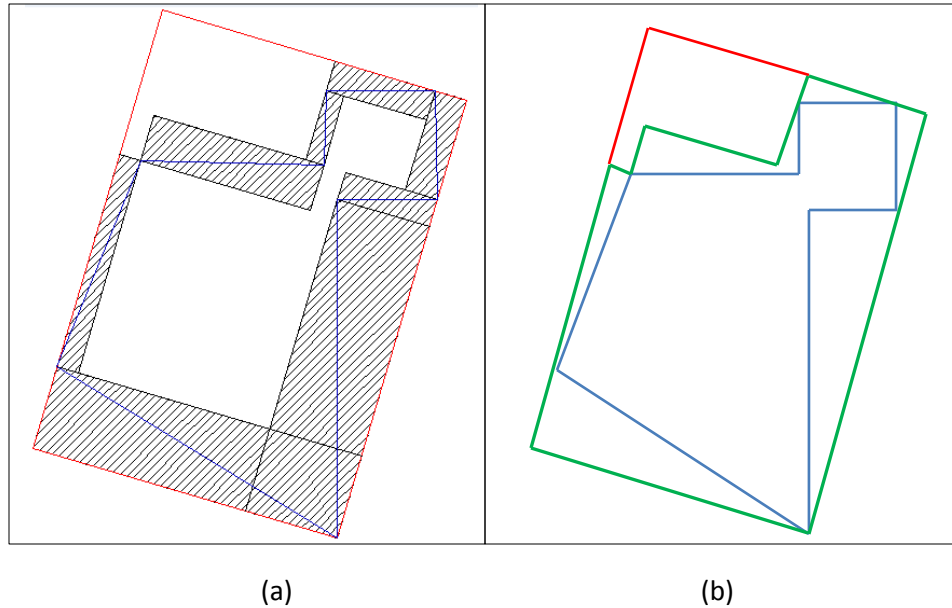


Figure 7.6 Squaring of an initial amalgam of a building cluster created using a synthetic data set with ear polygons: (a) initial amalgam in blue colour with triangular polygons (ears) hatched and (b) squared amalgam in green colour. Note: an amalgam with non -orthogonal edges has been chosen deliberately in order to enhance the visual impression of the squaring technique despite squaring being applied to amalgams with approximate orthogonal edges.

- iv. Merge all triangular polygons (ears) with the buffered amalgam to end up with a squared amalgam as shown in Figure 7.6(b).

Squaring algorithm II

- i. Follow steps (i) to (iv) of the squaring algorithm I and get the merged and squared amalgam.
- ii. Calculate the statistical wall orientation by Duchêne *et al.* (2003) of the squared amalgam and the orientation of the MBR and get the difference.
- iii. Rotate the squared amalgam by the orientation difference to get the final squared building amalgam with the rotation point selected as the centroid of the MBR or the buffered amalgam.

Squaring algorithm III

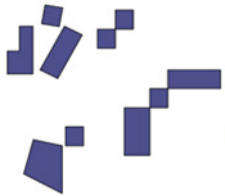
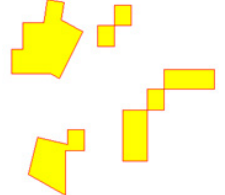
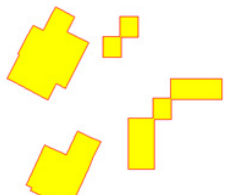
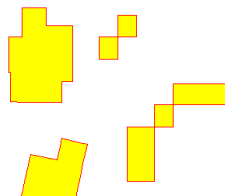
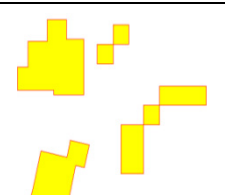
- i. Calculate the centroid of the buffered amalgam.
- ii. Calculate the maximum weighted local orientation (Duchêne *et al.*, 2003) of the buffered amalgam.
- iii. Rotate the amalgam to align its edge with the maximum local weighted orientation along the X-axis.
- iv. Calculate the MBB of the rotated amalgam oriented along the X-axis.
- v. Rotate back the MBB with a rotation equal to the previous value to get the GOBR.
- vi. Rotate the amalgam back as well and use this amalgam for subsequent ear polygon calculation as explained in the squaring algorithm I without using the original amalgam to maintain consistency between the GOBR and the buffered amalgam.
- vii. Calculate the orientations of both width and height of the GOBR.
- viii. Calculate the ear polygons based on these two orientations by iterating through each edge of the buffered amalgam.
- ix. Merge all the ear polygons to get the final squared amalgam.

See Appendix G.5 for the pseudo code of the algorithm.

7.2.4 Testing of amalgams after squaring edges

Table 7.1 depicts the results of cluster aggregation and squaring on a synthetic data set (the source) using the dilation and erosion algorithm I described in Section 7.2.1, coupled with the squaring algorithms I, II and III, each described in Section 7.2.3.

Table 7.1 Test results of the building aggregation algorithm with dilation and erosion followed by squaring the sides of the amalgam. Threshold value used: cluster distance (buffer distance): 10m.

Algorithm	Geometry	Visual representation	Criteria	
Aggregation with Squaring** comprising of contributing algorithms dilation and erosion algorithm I*, and Squaring algorithms I, II and III***	Source		Orientation	Shape
	Aggregated with buffer		Orientation of cluster outline preserved	Preserved, but corner touching buildings not aggregated
	Squared with Squaring algorithm I		Not preserved	Not preserved
	Squared with Squaring algorithm II		Preserved	Not preserved
	Squared with Squaring algorithm III		Preserved	Preserved

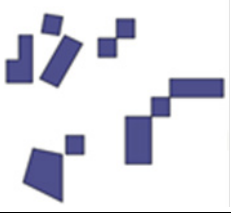
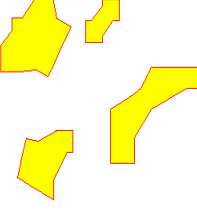
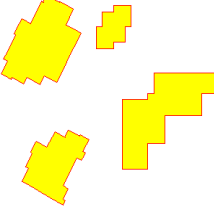
** Modification and/ or extension to an existing algorithm

* Existing algorithm

*** New algorithm

Table 7.2 depicts the results of cluster aggregation and squaring on a synthetic data set (source) using the concave hull generation algorithm described in Section 7.2.2, coupled with the squaring algorithm III described in Section 7.2.3.

Table 7.2 Test results of the building aggregation algorithm with concave hull generation followed by squaring the sides of the amalgam. Threshold values used: cluster distance (buffer distance) - 10m and hull creation percentage - 0.99.

Algorithm	Geometry	Visual representation	Criteria	
Aggregation with Squaring** comprising of contributing algorithms - Concave Hull generation* for aggregation and Squaring algorithm III***	Source		Orientation	Shape
	Aggregated		Orientation of cluster outline not preserved	Not preserved, but corner touching buildings aggregated
	Squared with Squaring algorithm III		Not preserved	Not preserved and many granular sides introduced

* Existing algorithm
 ** Modification and/or extension to an existing algorithm
 *** New algorithm

When analysing the results of the amalgams shown in Table 7.1, it is observed that the squaring algorithm III gives promising results in terms of orientation and shape preservation of the original clusters except the clusters with buildings in corner touching positions. However, the shape should be improved to deal with narrow sections (bottlenecks), juts and granular sides introduced as a result of the amalgamation and squaring. Further, squared amalgams created by the concave hull generation do not preserve both orientation and shape of the original clusters as shown in Table 7.2. When squaring, some granular sides would be introduced because the close interval vertices are formed on its outline at the time of creating the amalgam by the concave hull generation.

However, the generation of the concave hull has eliminated corner touching exceptions in the clusters when viewing the aggregated result in Table 7.2. As a result of these observations, it is understood that the amalgam creation before squaring should be further improved to deal with clusters of buildings in the corner touching and almost overhanging instances. For this purpose, the dilation and erosion with buffer operation II given in Section 7.2.1 is used with the squaring algorithm III for further testing with implementations, developing new algorithms to remove narrow sections (bottlenecks), juts and granular sides which may exist in the amalgam.

7.2.5 Enlargement of narrow sections and juts

Enlargement algorithm I

This algorithm is developed to apply to amalgams created by the two algorithms: the dilation and erosion with buffer operation II (see Section 7.2.1) and the squaring algorithm III (see Section 7.2.3).

- i. Calculate the centroid and the maximum weighted local orientation by Duchêne *et al.* (2003) of the squared amalgam.
- ii. Rotate the amalgam to align its edge with the maximum local weighted orientation along the X-Y axes.
- iii. Calculate the MBB of the rotated amalgam oriented along the X-Y axes.
- iv. Create inner line strings of the amalgam by extending each edge of its outline to meet the opposite edge as shown in Figure 7.7(b). Use the following algorithm for creating inner line strings.
 - a. Select each edge of the outline.
 - b. Extend the edge from its both ends until it meets and intersects the outline of the amalgam.

- c. Get these new extensions of the inner line strings and store them with the line strings of the outline of the amalgam in an array.

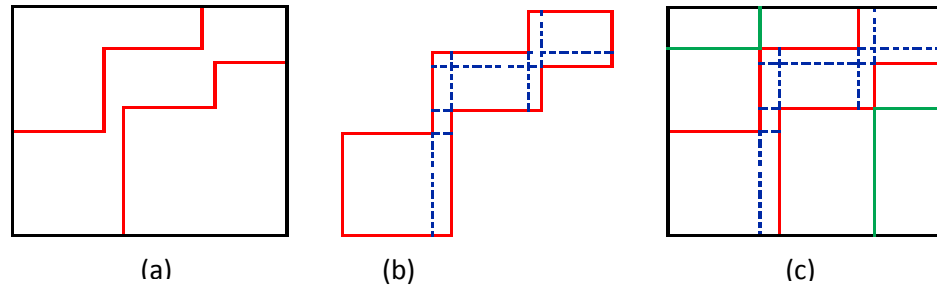


Figure 7.7 Creating rectangular strips required in the enlargement process: (a) squared amalgam with narrow corridors surrounded by the MBB (b) line string extensions inside the amalgam in broken lines by extending edges towards the respective opposite edges and (c) line string extensions inside polygons created by the difference of the squared amalgam from the MBB.

- v. Polygonize all the line strings in the array to create inner rectangular shape polygon strips as shown in Figure 7.7(b).
- vi. Similarly, polygonize all the line string extensions (shown in green) and the line strings of the outline of all the polygons created by obtaining the polygon difference between the amalgam and the MBB as shown in Figure 7.7(c) in order to create rectangular polygons between the region of amalgam and the MBB, which is called as the space region.
- vii. Sort the inner rectangular polygon strips with the use of the centroid along the X-axis and the Y-axis and store vertical inner polygons on a X_strip stack and horizontal inner polygons on a Y_strip stack separately.
- viii. In the case of stacking X_strips, if the width of the strip is larger than the bottleneck threshold of the buffered amalgam, such strips are not stacked and ignored. Similarly, in the case of Y_strips, if the height of the strip is larger than the bottleneck threshold, such strips are not stacked and ignored.
- ix. Iterate through the X_strip and the Y_strip stacks separately and union the polygons in each X_strip and Y_strip and store in two stacks (X_stack and Y_stack) separately.

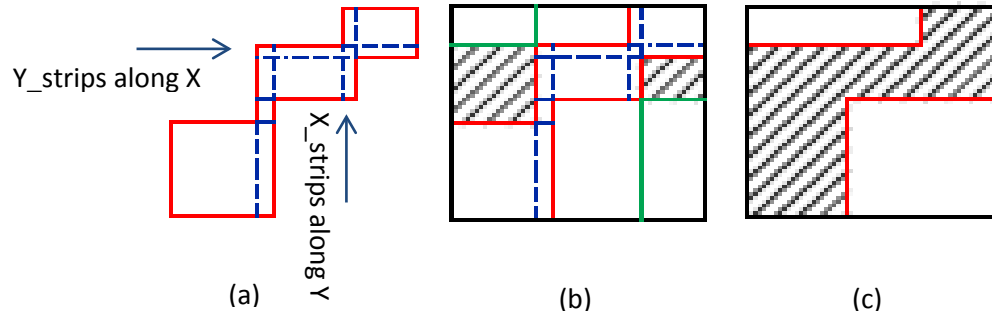


Figure 7.8 Selection of filling rectangular strips in the enlargement process: (a) polygon strips along the X-Y axes (b) chosen polygon rectangles between the squared amalgam and the MBB (space region) to fill the bottlenecks of the amalgam and (c) final squared amalgam hatched after filling operation.

- x. Iterate through each union polygon on the X_stack and identify the left and the right polygon rectangles in the space region that share a line with the union strip with the help of the MBB coordinates of both the union polygon and the space polygon rectangles.
- xi. Similarly, identify the top and the bottom polygon rectangles in the space region that share a line with the union strips in Y_stack with the help of the MBB coordinates.
- xii. In both (x) and (xi) steps, if two space rectangular polygons are selected on either side of a union strip, the rectangular polygon with the minimum area out of the two polygons is selected as one of the candidate filling polygons and stored in an array. If both areas are equal, the top polygon is selected from the Y_stack and the right polygon is selected from the X_stack to represent the filling polygons. During this operation, if the same space rectangular polygon is selected with reference to both axes, such duplicates are removed from the array list consisting of filling space rectangular polygons.
- xiii. Merge all the filling polygons thus obtained with the squared amalgam to come out with the squared and enlarged building amalgam as shown in Figure 7.8(c).

- xiv. Finally, if the initial amalgam before squaring has been created with the iterative erosion procedure as explained in the dilation and erosion with buffer operation II (Section 7.2.1), apply a negative buffer with the distance left for total erosion to the squared and enlarged amalgam to have the final refined amalgam.
- xv. Rotate the refined amalgam back to its original orientation.

Enlargement algorithm II

Enlargement algorithm II is the modified version of the enlargement algorithm I.

- i. Adapt the same steps i to xi explained in the enlargement algorithm I.
- ii. In both steps x and xi, if two or more space rectangular polygons are selected on either side (either polygons in top-bottom combination or right-left combination) of a single strip while iteration, filling of the strip is performed evenly on either side with the given filling threshold value (Figure 7.9).
- iii. In cases where only the single sided space polygons are selected (e.g. left space polygons are selected while no polygons are selected on the right side of the strip), filling of the selected polygons is performed with the threshold width if and only if the strip boundary adjoins the boundary of the MBB, if not, ignore filling that particular strip (Figure 7.9(d)).

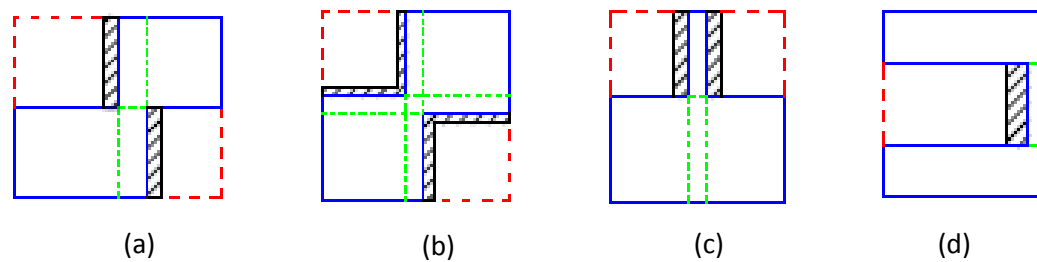


Figure 7.9 Possible cases of building enlargement with narrow sections: (a) enlargement of both sides of the narrow section of a building comprising of a pair of opposite concave corners aligned along the X-Y axes (b) enlargement based on the narrow sections oriented on the X-Y axes with the pair of opposite concave corners aligned diagonally (c) enlargement of both sides of a jut and (d) single-sided enlargement of a narrow section.

- iv. Merge all the filling polygons thus obtained with the squared amalgam to create the squared and enlarged building amalgam as shown in Figure 7.8(c).
- v. Finally, if the initial amalgam before squaring has been created with the iterative eroding procedure as explained in the dilation and erosion with the buffer operation II (Section 7.2.1), apply a negative buffer with the distance left for total erosion to the squared and enlarged amalgam to have the final refined amalgam.
- vi. Rotate the refined amalgam back to its original orientation.

7.2.6 Simplification of granular edges

The squaring and the enlargement algorithms produce granular edges comprising of pseudo corners (corners that do not exist in the source data) in the building outline and requires removal of such corners in addition to the building edges with a length less than the minimum building edge distance, depending on the target scale (Figure 7.10). Simplification of building edges is required for this purpose.

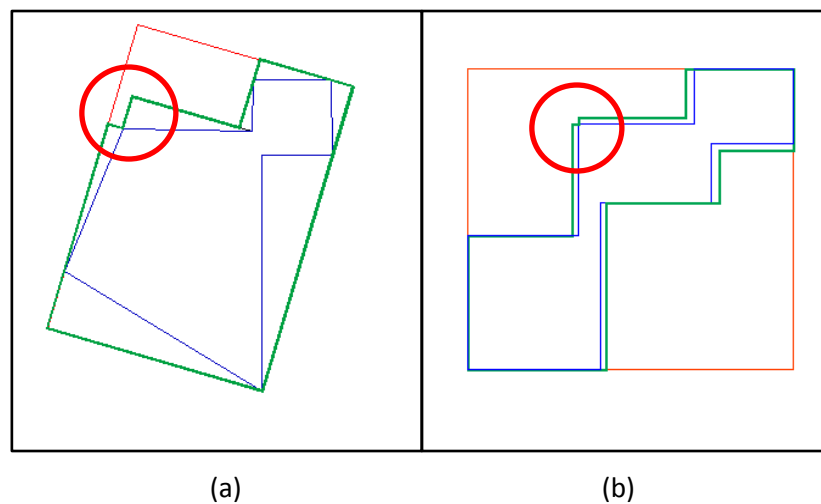


Figure 7.10 Introduction of granular edges in the amalgam after squaring and enlargement: (a) pseudo concave shape formed (circled) at a convex corner due to building squaring and (b) a similar pseudo concave shape due to the enlargement of narrow sections of a building.

Simplification algorithm I

- i. Calculate the centroid and the maximum weighted local orientation by Duchêne *et al.* (2003) of the squared and enlarged amalgam.
- ii. Rotate the amalgam and align it along the X-Y axes with the maximum local weighted orientation.
- iii. Iterate through edges of the amalgam and find each concave vertex with the two other immediately attached vertices.
- iv. Find the MBB of these three vertices by iterating through each concave vertex. The MBBs thus created are the candidate filling polygons to carry out simplification.
- v. Filter out the potential filling polygons based on the minimum threshold distance in the simplification process.
- vi. Glue such polygons with the amalgam oriented along the X-Y axes.
- vii. Get the new amalgam and run the steps III to VI iteratively until no filling polygon is found with the edges less than the simplification threshold to get the combined amalgam.
- viii. Rotate this amalgam back to its original orientation to get the final simplified amalgam.

When analysing the output geometrically within the implementation in Java, the filling polygons created do not exactly fit with the amalgam to which the filling polygons are to be glued because the orientation of the rotated amalgam does not get exactly oriented along the X-Y direction due to the angular precision value (0.25^0 degrees) used in the orientation calculation algorithm by Duchêne *et al.* (2003) implemented in this research. Therefore, an improved simplification algorithm is required and dealt with next.

Simplification algorithm II

- i. Adopt the same steps I and ii in the simplification algorithm I.
- ii. Create the space regions between the MBB and the amalgam oriented along the X-Y axes.
- iii. Create line strings inside space regions by extending them until they meet the MBB and polygonize those to form space polygons as shown hatched in Figure 7.11(b).

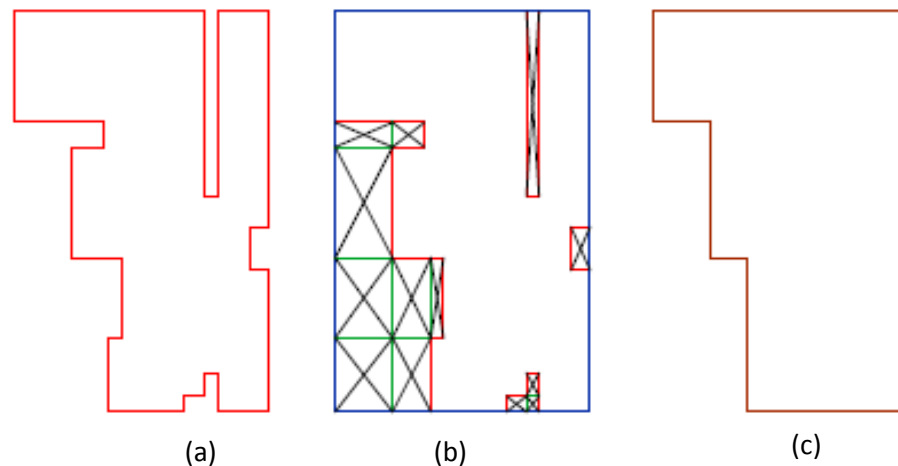


Figure 7.11 Simplification of the amalgam with orthogonal sides: (a) original building amalgam oriented along the X-Y axes (b) space polygons (rectangles) between the MBB and the original building amalgam along the X-Y axes hatched with a cross and (c) simplified building amalgam with a simplification tolerance of 5.0m.

- iv. Iterate through each vertex of the amalgam and find the concave vertices using the exterior angle between two edges connecting each vertex of the amalgam. In this process, if two adjoining concave vertices are found, one vertex is ignored.
- v. Iterate through each concave vertex and filter out each space polygon (rectangle) on which a concave vertex lies. If one of the sides of such a polygon is below the simplification threshold value, select and store it as one of the candidate polygons to glue with the amalgam.
- vi. Glue all the filtered polygons (rectangles) with the amalgam after completion of the iteration of concave vertices.

- vii. Get the new amalgam and run steps (ii) to (vi) iteratively until no space polygon is found with the edges less than the simplification threshold to get the combined and simplified amalgam as shown in Figure 7.11(c).
- viii. Rotate this amalgam back to its original orientation to get the final simplified amalgam.

See Appendix G.7 for the pseudo code of the algorithm.

7.2.7 Testing of amalgams after complete generalization process

In this phase, the main generalization algorithm of building aggregation with orthogonal sides is tested and evaluated while testing and evaluating intermediate algorithms developed for enlargement of narrow sections and juts, and simplification of granular sides. These two intermediate algorithms are applied to the squared amalgams obtained with the two algorithms - dilatation and erosion with the buffer operation II and the squaring algorithm III. Figure 7.12 depicts the functional model of the algorithm.

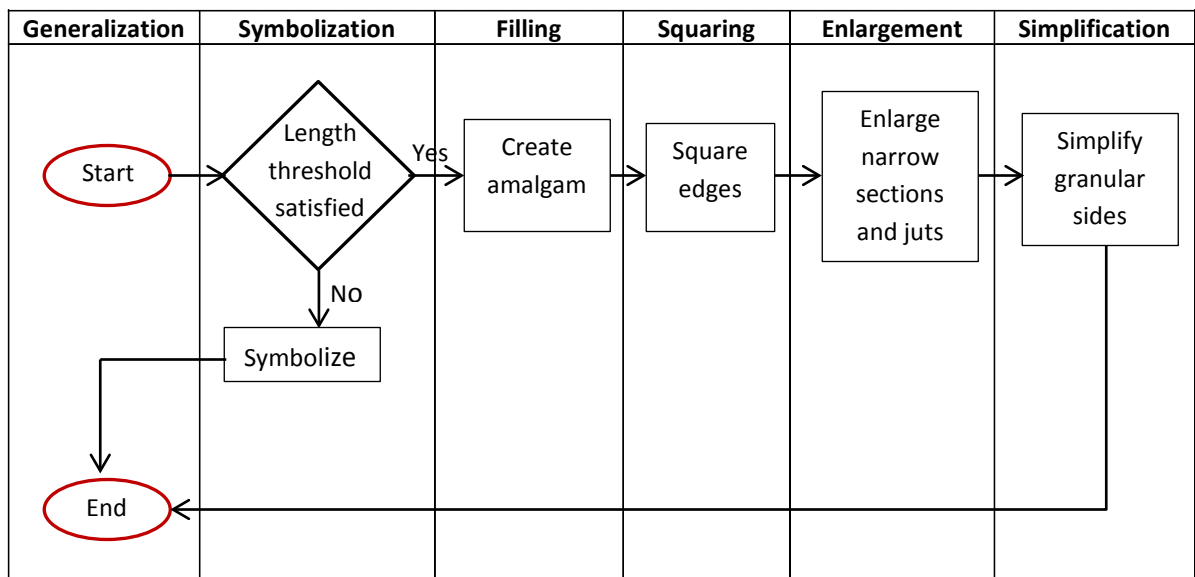
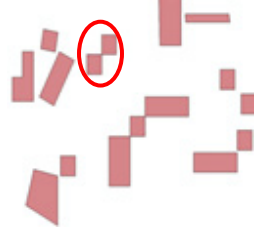
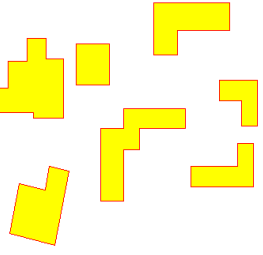
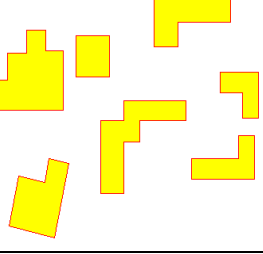


Figure 7.12 Functional process of building symbolization and aggregation with orthogonal sides.

Table 7.3 illustrates the test results of the building aggregation algorithm with orthogonal sides on a synthetic data set using a combination of the dilation and erosion algorithm with buffer operation II (see Section 7.2.1), the squaring algorithm III (see Section 7.2.3), the enlargement algorithm I (see Section 7.2.5) and the simplification algorithm II (see Section 7.2.6).

Table 7.3 Test results I of building aggregation algorithm with orthogonal sides. Threshold values used: cluster distance (buffer distance) - 10m, Minimum building length - 15m, Minimum width of narrow sections - 5m and simplification tolerance - 3m. The cluster circled in red colour is symbolized by a rectangle using the symbolization algorithm in Section 7.1.

Algorithm(s)	Geometry	Visual representation	Criteria	
Symbolization *** and aggregation algorithm comprising of contributing algorithms - dilation and erosion with buffer operation II **, squaring algorithm III **, enlargement algorithm I ** and simplification algorithm II **	Source		Orientation	Shape
	Symbolized or aggregated, squared and enlarged		Orientation of the clusters preserved	Not preserved in corner touching, almost overhanging and total overhanging clusters
	Symbolized or aggregated, squared, enlarged and simplified		Orientation of the clusters preserved	Not preserved in corner touching, almost overhanging and total overhanging clusters

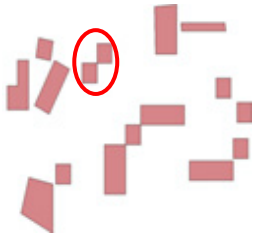
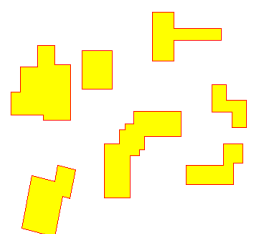
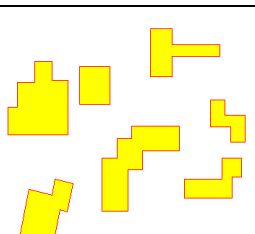
** Modification and/or extension to an existing algorithm

*** New algorithm

When analysing the test results, it is observed that the aggregated results do not preserve the shape very well in exceptional clusters, as in the case of buildings located touching at the corner, almost overhanging or total overhanging positions despite creating an amalgam, removing corner touching and hanging situations with the dilation and erosion with buffer operation II. Therefore, the enlargement algorithm dealing with filling up narrow sections and juts needs to be further modified and tested for improved shape preservation of the amalgam. The cluster with the rectangular corner touching pair of buildings is only symbolized using the algorithm described in Section 7.1 since it does not meet the legibility constraint - minimum edge length - to be represented in the target scale.

Table 7.4 illustrates the test results of the building aggregation algorithm with orthogonal sides on the same synthetic data set used in Table 7.3 using the same combination of algorithms except the algorithm for the enlargement which is replaced by a modified algorithm of the previous enlargement algorithm I, named as enlargement algorithm II (see Section 7.2.5). When analysing the test results, it is observed that the modified enlargement algorithm preserves shape characteristics of the source clusters. The final results of the aggregation algorithm with orthogonal sides are satisfactory in preserving both orientation and shape characteristics in dealing with any exceptional case of building locations which would rarely come across in real data sets. However, thorough testing of these types of exceptional cases needs to be done for further verification of the algorithm.

Table 7.4 Test results II of the building aggregation algorithm with orthogonal sides. Threshold values used: cluster distance (buffer distance) - 10m, Minimum building length - 15m, Minimum width of narrow sections - 5m and simplification tolerance - 3m. The cluster circled in red colour is symbolized by a rectangle using the symbolization algorithm in Section 7.1.

Algorithm(s)	Geometry	Visual representation	Criteria	
Symbolization ^{***} and aggregation algorithm comprising of contributing algorithms - dilation and erosion with buffer operation II ^{**} , squaring algorithm III ^{***} , enlargement algorithm II ^{***} and simplification algorithm II ^{***}	Source		Orientation	Shape
	Symbolized or aggregated, squared and enlarged		Orientation of the clusters preserved	Preserved in corner touching, almost overhanging and total overhanging clusters, but granular sides introduced
	Symbolized or aggregated, squared, enlarged and simplified		Orientation of the clusters preserved	Preserved in corner touching, almost overhanging and total overhanging clusters

^{**} Modification and/or extension to an existing algorithm

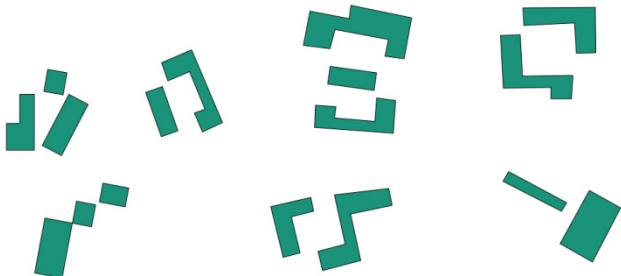
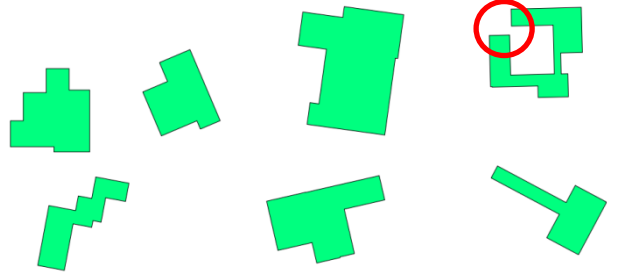
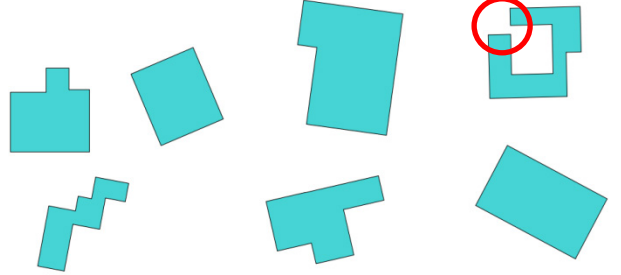
^{***} New algorithm

Further testing of the algorithm with some exceptional synthetic data

Further testing on another exceptional synthetic data set reveals that the algorithm collapses. This is evident when observing the amalgamated result circled in red colour on the building cluster shown in Table 7.5 where one hanging corner (upper) of the cluster has not been bridged. Analysing further, it is found that the above exception occurs when the polygon does not become a multi-polygon in the iterative buffer process to terminate its execution under the dilation and erosion with buffer operation II. The reason is that the algorithm keeps on iterating with the amalgam yet has been treated to be a polygon using the bridge on the other side of the hanging corner (down) until the bridge gets collapsed.

This occurs when the distance between the buildings in the non-collapsing hanging position is greater than the distance in the other collapsing hanging position. Therefore, the algorithm needs to be revised to handle any type of data consisting of simple data to very complex data of building shapes.

Table 7.5 Results of further testing of the building aggregation algorithm with orthogonal sides using an exceptional synthetic data set. No symbolization is applied according to the value of the length threshold used in the testing.

Type of data	Distance thresholds in meters (m)	Visual representation
Source building data in the form of clusters	Clustering: 8.5	
Amalgams after aggregation, squaring and enlargement of juts	i. Length threshold for symbolization: 20 i. Dilation and erosion: 8.5 ii. Enlargement:5	
Final amalgams after aggregation, squaring, enlargement of juts and simplification of edges	i. Length threshold for symbolization: 20 ii. Dilation and erosion: 8.5 iii. Enlargement:5 iv. Simplification:5	

7.2.8 Modified building aggregation algorithm with orthogonal sides

The dilation and erosion algorithm with buffer operation II described in Section 7.2.1 collapses in creating the initial amalgam of a cluster with the filling operation before applying squaring, enlargement and simplification operations. Therefore, the previous algorithm is modified by introducing a solution to fill in the gap space between buildings inside the cluster to create the initial amalgam using 2D spatial triangulation. The other algorithms developed and tested for squaring, enlargement and simplification are incorporated in this modified algorithm. However, the inner holes of polygons if available, are not treated in this algorithm. The algorithm is explained in detail below.

Algorithm

- i. Read a non-single cluster of orthogonal shape from the PostGIS database.
- ii. Run the symbolization algorithm described in Section 7.1 on the cluster.
- iii. If the threshold length applied to the building edges according to target scale is greater than the minimum length of the bounding rectangle created in the symbolization algorithm, symbolize the cluster and go to step (w) or else go to the next step.
- iv. Apply buffer operation to the cluster using the dilation and erosion algorithm I described in Section 7.2.1.
- v. Check if the buffered amalgam is a polygon. If it is a polygon, adapt the following steps.
 - a. Apply the squaring algorithm III to square the edges of the buffered amalgam. In the application of this algorithm, use the wall orientation of the union building cluster to get the GOBR because the wall orientation can be changed after gluing the buildings.
 - b. Apply the enlargement algorithm II on the squared amalgam.

- c. Apply the simplification algorithm II (Section 7.2.6) on the enlarged amalgam to get the final amalgam and go to step (w).

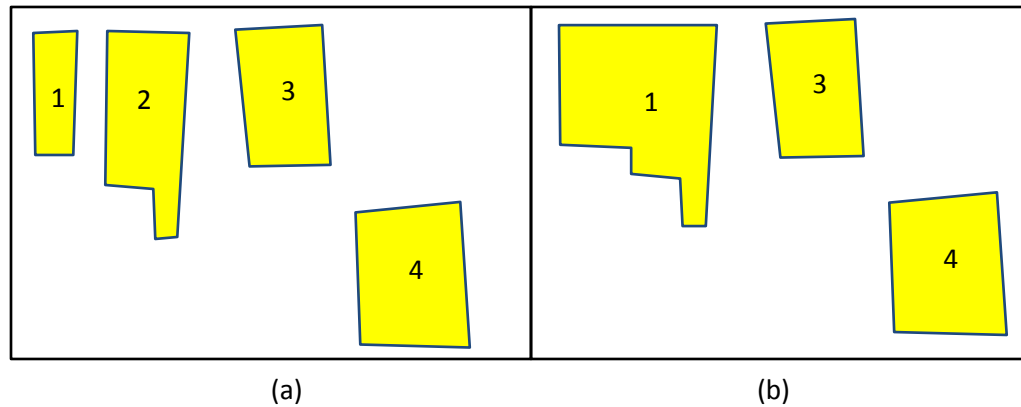


Figure 7.13 Result of buffering with same positive and negative distance: (a) source cluster and (b) buffered amalgam with the two leftmost buildings (IDNs 1 and 2) bridged together, and two other source buildings (IDNs 3 and 4) in the cluster.

- vi. If it is a multi-polygon, it can have disjoint polygons which may consist of either original source polygons in the cluster (all polygons with corner touching and/or hanging locations) or a combination of glued polygons due to dilation and erosion and source polygons or only the glued polygons located apart (e.g. buildings 3 and 4 are in an overhanging position as depicted in Figure 7.13). Also, get the wall orientation to be used in squaring before gluing buildings (wall orientation of the multi-polygon). Then, adapt the following steps for creating the amalgam.
 - a. Reassign dummy IDNs to each polygon in the buffered amalgam geometry.
 - b. Triangulate the buffered amalgam with the DCT developed in this research (see Section 4.4) as shown in Figure 7.14.

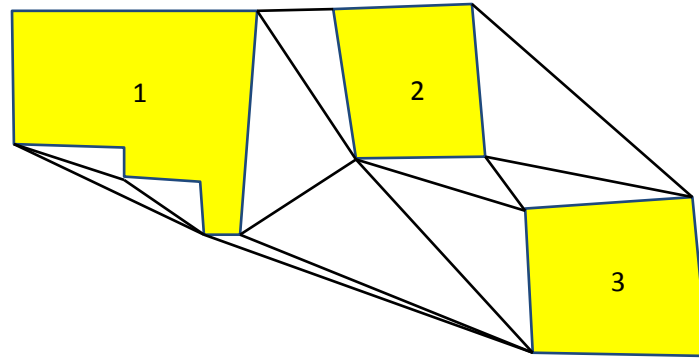


Figure 7.14 DCT on the buffered amalgam.

- c. Get the adjacent links and work out the minimum distance between each pair of buildings and get a list of links in the form of [bid1, bid2, distance].
- d. Iterate through the list, get each pair of buildings and do the following if the minimum distance between each pair is less than or equal to the clustering threshold.
- e. Retrieve geometries of the two buildings in the pair.
- f. Run the CNDT with constraint edges on the pair of buildings. The purpose of running the CNDT is to have more hooks between the two buildings in the pair. This will enable to have the least exaggeration of the gap between the two buildings.
- g. Retrieve the source building IDNs attached to each triangle in the CNDT.
- h. With building IDN information, find out the space triangles between the two buildings and assign them to an array (triangles that connect the two buildings only). In this case, all the building triangles and the false triangles (see Figure 4.11 in Section 4.4.2) are omitted.
- i. Iterate through the array of space triangles and get each triangle.
- j. Now choose triangles to bridge the gap between each pair of buildings so that at least an edge of each space triangle, which connects both buildings, is less than

or equal to the distance threshold used in forming building clusters (see highlighted triangles in Figures 7.15(a) and 7.15(b)).

- k. These triangles are glued to obtain a single geometry (highlighted multi-polygons in yellow colour in Figure 7.15).
- l. Separate glued polygons if multi-polygon geometry exists (two polygons each in Figures 7.15(a) and 7.15(b)).
- m. Iterate over each separated polygon and retrieve the triangles consisting of each such polygon (each separated polygon needs to be buffered with a small positive distance to track triangles within each such polygon).
- n. If such a polygon consists of a single triangle or a pair of triangles, such triangles are added to the final array of bridging triangles (e.g. triangle 1, 2 and 3 in Figure 7.15(a)).
- p. If such a polygon consists of more than two triangles (e.g. triangles 1, 2 and 3 in Figure 7.15(b)), map shared edges of each triangle with its edge lengths and retrieve the pair of triangle IDNs which has the shared edge of the minimum length and add into the final array of bridging triangles (triangles 2 and 3).

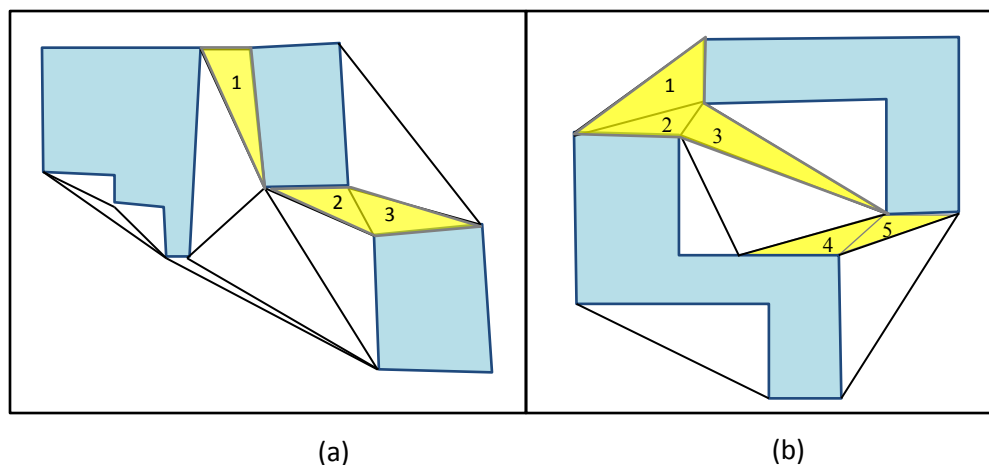


Figure 7.15 Triangles selected for bridging based on the distance threshold highlighted in yellow colour.

- q. Glue the finally selected triangles with the source buildings to get the initial building amalgam. Check if the amalgam is a polygon. If it is a polygon, adapt sub-steps (a) to (c) of step (v).
- r. If it is a multi-polygon, apply a small positive buffer (say distance - Δd) to make the entire amalgam a polygon.
- s. Now adapt sub-steps (a) and (b) under step (v) for squaring and enlarging the amalgam.
- t. Apply a negative buffer with the same distance (Δd) used for positive buffering in step (r) above.
- u. Adapt sub-step (c) under step (v) to simplify the edges to have the final amalgam cluster.
- w. Go to step (i) iteratively until the end of all the clusters.

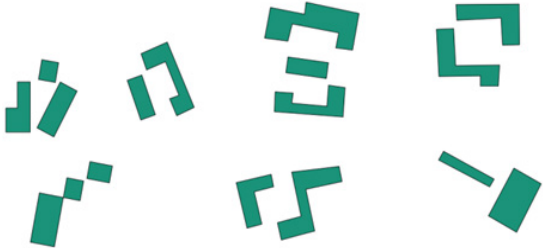
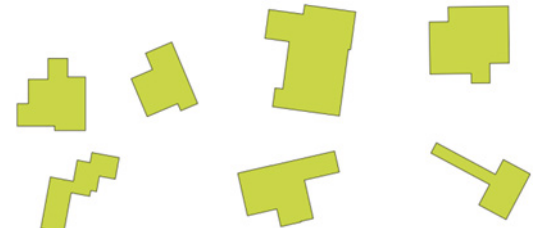
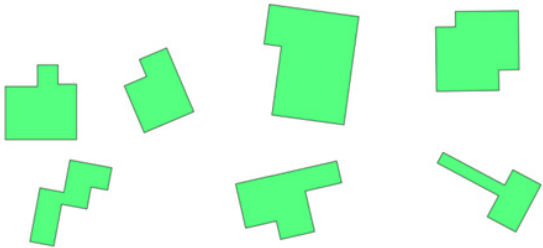
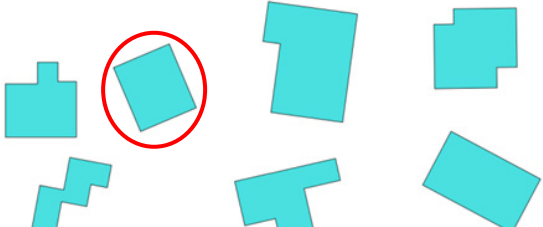
7.2.9 Testing of amalgams with the modified algorithm

With a synthetic data set

When comparing the results of the amalgams in Table 7.6 with the results (see Table 7.5) obtained from further testing of the previous algorithm in Section 7.2.7, it can be seen that the cluster at the top right-hand corner, which could not be amalgamated properly due to the exception in dilation and erosion with the buffer operation II in the previous algorithm, is now amalgamated correctly.

Further, it is observed at the bottom Figure in the Table 7.6 that the algorithm has applied symbolization (circled in red colour is a symbolized amalgam) as described in Section 7.1 instead of aggregation with the squaring operation when increasing the length threshold. The algorithm has the capability to handle buildings in exceptional places such as corner touching and overhanging positions in clusters.

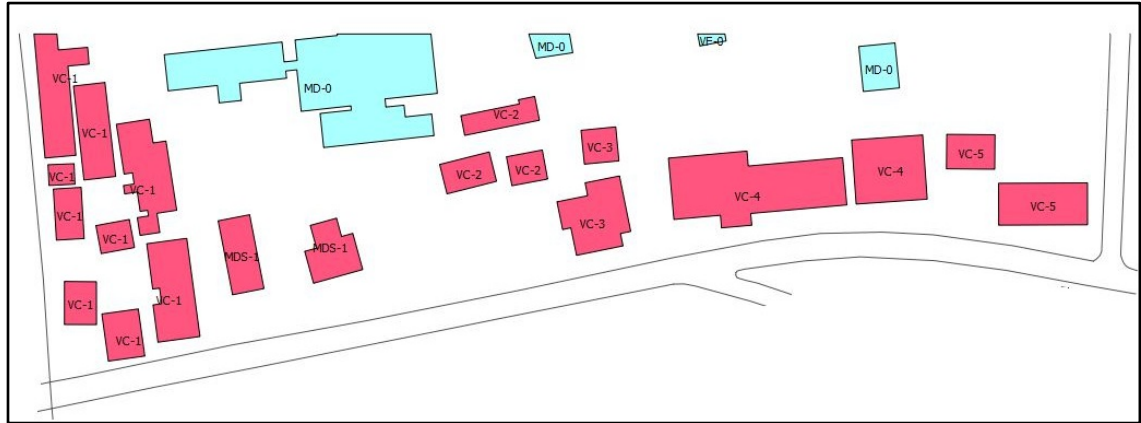
Table 7.6 Results of the aggregation of building clusters with orthogonal sides on the synthetic data used in Table 7.5 with some exceptional building configuration. Circled in red colour is a symbolized amalgam using the symbolization algorithm in Section 7.1.

Geometry	Distance threshold in meters (m)	Visual representation
Source building data in the form of clusters	Clustering - 8.5	
Amalgams after symbolization or aggregation, squaring and enlargement of juts	i. Length threshold for symbolization** - 20 ii. Dilation and erosion* - 8.5 iii. Enlargement** - 5	
Final amalgams after symbolization or aggregation, squaring, enlargement of juts and simplification of edges	i. Length threshold for symbolization** - 20 ii. Dilation and erosion* - 8.5 iii. Enlargement** - 5 iv. Simplification** - 5	
	i. Length threshold for symbolization** - 25 ii. Dilation and erosion* - 8.5 iii. Enlargement** - 5 iv. Simplification** - 5	

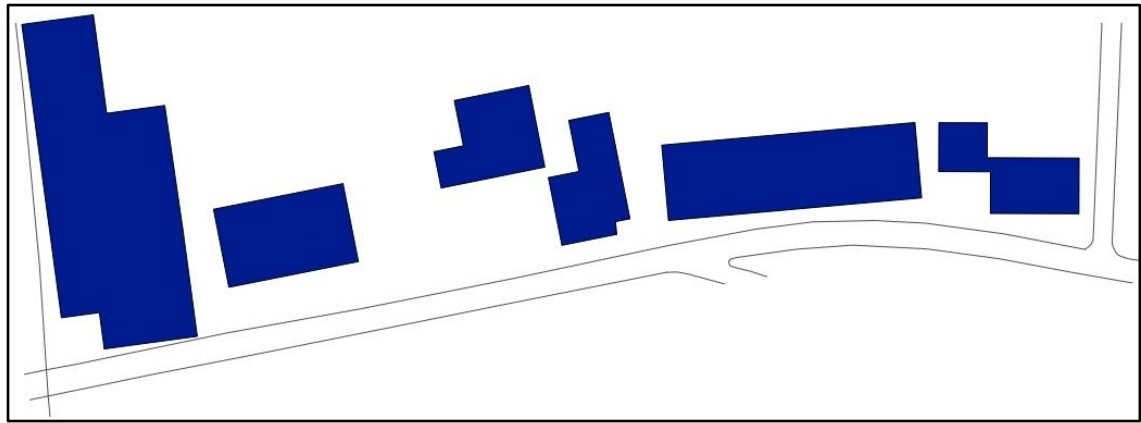
** new algorithm

* Existing algorithm

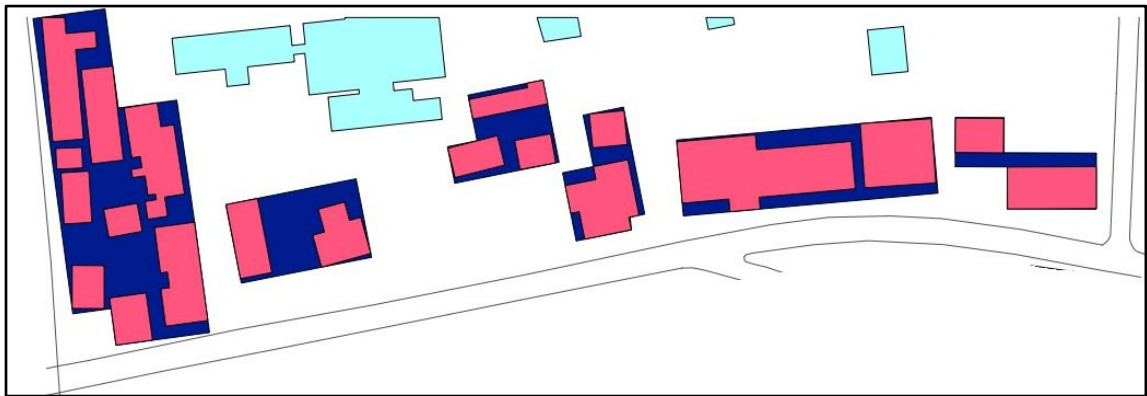
With a real data set



(a)



(b)



(c)

Figure 7.16 Results of the aggregation of building clusters with orthogonal sides on a real data set representing a region after squaring, enlargement and simplification. Thresholds: clustering, enlargement and simplification - 5m and building side length - 20m (a) clusters in red (b) generalized buildings and (c) results overlaid with the source data. Data source: The NMA of Sri Lanka. Copyright reserved.

Figure 7.16 depicts the results of the amalgams with an orthogonal outline, created by aggregating buildings in each cluster. The implemented algorithm is applied to the clusters belonging to ‘very close’ and ‘medium range’ classifications (Figure 7.16(a)). The two rectangles shown in the generalized result are produced by the symbolized algorithm based on the building side length threshold.

Synopsis of the contributing algorithms used

Table 7.7 below summarises the contributing algorithms of the main cluster aggregation with orthogonal sides.

Table 7.7 Building cluster aggregation algorithm with orthogonal sides, including contributing algorithms.

Main algorithm	#	Contributing algorithm	Purpose
Building cluster aggregation with orthogonal sides	1	Symbolization ^{***}	To Exaggerate the size of building geometry
	2	Dilation and erosion with buffering algorithm [*]	To fill the gaps between building geometries
	3	DCT ^{**}	To get adjacency relations between building geometries
	4	CNDT [*]	To fill the gaps between buildings with triangles
	5	Squaring ^{***}	To square edges of the amalgam
	6	Enlargement ^{***}	To enlarge narrow sections and juts in the amalgam
	7	Simplification ^{***}	To delete shorter edges of the amalgam

^{*} Existing algorithm

^{**} Modification and/or extension to an existing algorithm

^{***} New algorithm

7.3 Building cluster aggregation with non-orthogonal sides

The use of this algorithm is to create an amalgam by aggregating buildings in a non-orthogonal shaped cluster which is identified and enriched during the cluster shape enrichment process described in Section 5.4 under the data enrichment process. In developing this algorithm, the treatment of buildings in exceptional positions as explained in the development of aggregation of buildings with orthogonal sides is considered with the use of the CNDT with constraint edges (see Section 4.2) and edge adjacency relations of interconnected polygons to fill in the gaps between buildings.

7.3.1 Aggregation algorithm with triangulation

- i. Read a non-single cluster of non-orthogonal shape from the PostGIS database.
- ii. Run the symbolization algorithm described in Section 7.1 on the cluster.
- iii. If the threshold length of the building edges given according to the target scale is greater than the minimum length of the bounding rectangle created in the symbolization algorithm, symbolize the cluster and go to step (xii) or else go to the next step.
- iv. Apply buffer operation to the cluster using the dilation and erosion algorithm I described in Section 7.2.1. This operation may have created buildings that have got buffered forming self-connecting bridges due to concave shape (see Figure 7.17(b)) and the buildings that are partially glued in the form of a multi-polygon (see Figure 7.18(b)).

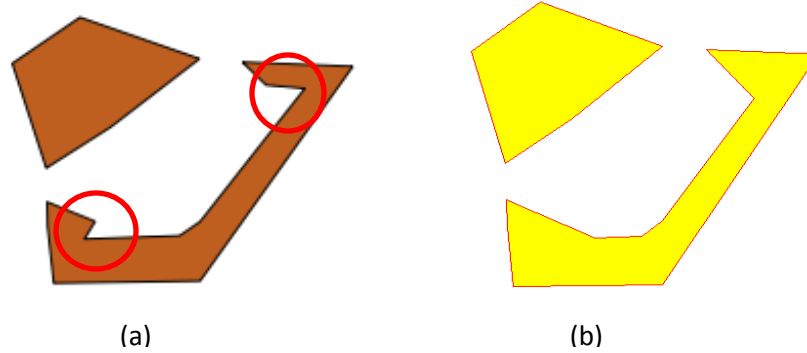


Figure 7.17 Result of buffering with the same positive and negative distance: (a) source cluster with the lower building comprising of two concave corners shown circled and (b) buffered amalgam where the two concave corners are filled as a result of the buffering operation.

- v. Carry out the following process to avoid self-bridging corners as explained in the previous step.
 - a. Get the filling bridges (if available) between each pair of buildings in the cluster and the self-filling bridges of single buildings (if available) by subtracting buffered amalgam from the union of the source building cluster.
 - b. Select only the bridges that connect two or more buildings in the cluster, ignoring self-filling bridges by iterating through each bridge.

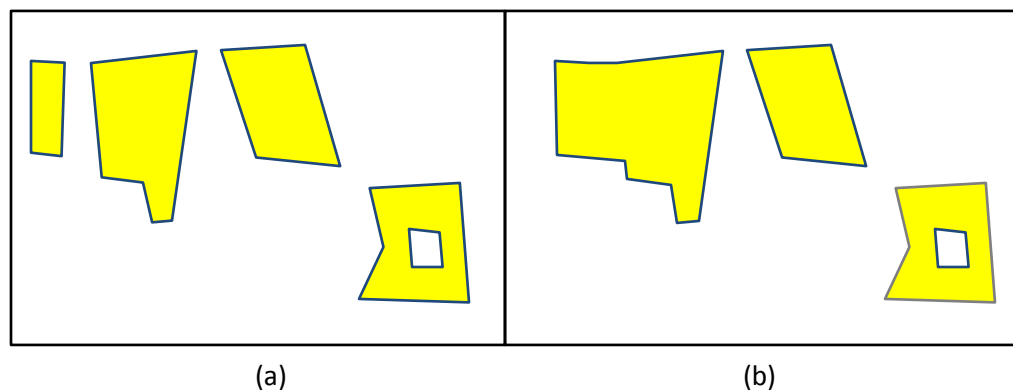


Figure 7.18 Result of buffering with the same positive and negative distance: (a) source cluster including a polygon with a hole and (b) buffered amalgam in a multi-polygon with a filling bridge between the two leftmost buildings and the preserved hole of the rightmost building in the source cluster in (a).

- c. Union all these bridges with the union of source building cluster. This operation preserves outline and inner holes of each building while bridges between buildings are formed (Figure 7.18(b)).
- vi. If the buffered amalgam is a polygon as a result of the aggregation of the cluster, go to step (xii). If the buffered amalgam is a multi-polygon as shown in Figure 7.18 (b), go to the next step.
- vii. Reassign dummy IDNs to each polygon in the buffered amalgam geometry.
- viii. Triangulate buffered amalgam with the DCT developed in this research (see Section 4.4) as shown in Figure 7.19.

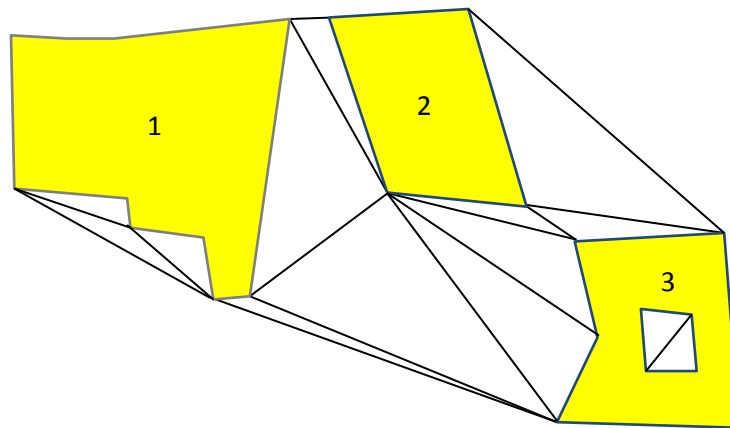


Figure 7.19 DCT on the buffered amalgam.

- ix. Get the adjacent links and work out the minimum distance between each pair of buildings and get a list of adjacency links in the form of [bid1, bid2, distance].

- x. Iterate through the list, get each pair of buildings and adopt the following if the minimum distance between each pair is less than or equal to the clustering threshold.
 - a. Retrieve geometry of both buildings in the pair.
 - b. Run the CNDT with constraint edges on the pair of buildings. The purpose of running the CNDT is to have more hooks between the pair of buildings. This will enable to have the least exaggeration of the gap between the two buildings.
 - c. Retrieve the source building IDNs attached to each triangle in the CNDT with constraint edges.
 - d. With building IDN information, find out the space triangles between the two buildings and assign them to an array (triangles that connect the two buildings only). In this case, all building triangles and false triangles (see Figure 4.11 in Section 4.4.2) in the space between the two buildings are omitted.
 - e. Iterate through the array of space triangles and get each triangle.
 - f. Find the three edges and normalise each edge to help extract a common pattern of all line segments with the use of their coordinates.
 - g. Keep a mapping between each normalised edge and the connecting triangle IDN (dummy IDN is used for each triangle as stored in the array) over iteration.
 - h. Find out topologically adjacent pairs of triangles from the map, which share an edge by finding all edges that appear exactly twice in the map (e.g. Pairs [1, 2], [2, 5], [3, 4] and [5, 3] (IDNs in red colour) in Figure 7.20).
 - i. Sort adjacent triangle IDNs in order with the help of adjacent pairs of triangle IDNs (e.g. IDNs 1, 2, 5, 3 and 4 in red colour in Figure 7.20) to get topologically adjacent triangles in an array.

- j. Reassign serial dummy IDNs to each triangle in the array, starting from IDN 1 and map these IDNs with the triangle geometry (dummy IDNs are numbered in black colour in Figure 7.20).
- k. Now by inputting the clustering distance and the space triangle edge distance threshold values, select the candidate space triangles where at least the distance of a single edge or more edges is less than or equal to the space triangle edge distance threshold (e.g. Figure 7.20 depicts selected space triangles highlighted in yellow colour) and get their initial triangle IDNs to bridge the gap between a pair of buildings. A wider filling bridge can be achieved by increasing the space triangle edge distance threshold.

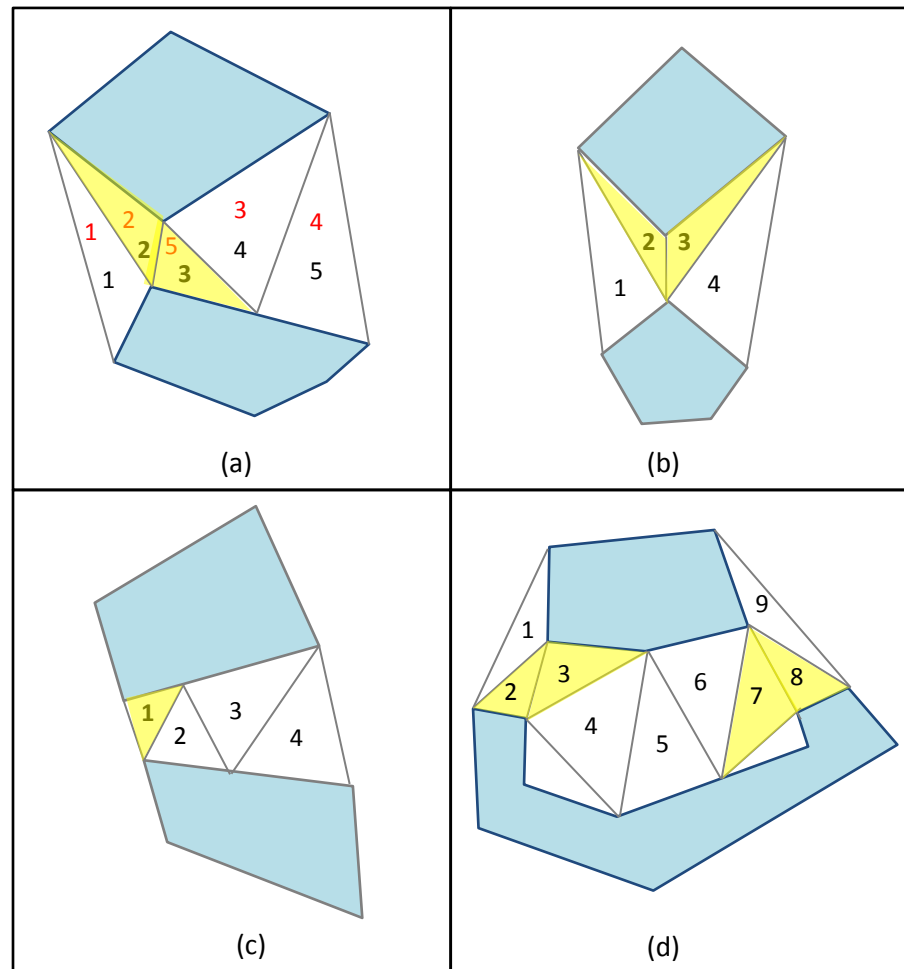


Figure 7.20 Possible cases of bridging a pair of buildings: (a) selection of an inverted pair of triangles (b) hanging selection with a non-inverted pair (c) selection of a single triangle and (d) selection of a couple of pairs of triangles in scattered locations.

- xi. Figure 7.20 illustrates the typical cases of bridging triangles that might occur due to the shape and the location of each building in the pair and the space triangle edge distance threshold. Apply the following steps for final selection of bridging triangles.
 - a. Apply pairing on the initially selected triangles (triangles highlighted in yellow colour in Figure 7.20) based on the edge distance threshold. There are seven cases that can cause triangle selection (see Table 7.8 for the pairing algorithm of adjacent triangles) based on pairing triangles.
 - b. If only one triangle IDN is selected (Figure 7.20(c)), it occurs at the edge of the convex hull. Iterate through topologically ordered and paired triangles until it gets an inverted pair (e.g. triangle IDNs 1 and 2 in Figure 7.20(c)) either in ascending order or descending order based on the first or the last IDN selected (see Table 7.8 for pairing) and get the triangle IDNs of the finally selected triangles.
 - c. If the triangle IDNs selected on pairing are serial with only a couple of triangles, check if the pair is either invertible (Figure 7.20(a)) or non-invertible (Figure 7.20(b)). If the pair is invertible, choose it as the finally selected pair of triangles for bridging. If it is non-invertible, get the two adjacent triangles on either side of the pair based on the pairing algorithm (see Case 3 in Table 7.8) and choose the triangle with the minimum area and add to the non-inverted pair to have the final selection of triangles (e.g. triangle IDNs 1, 2 and 3 in Figure 7.20(b)) for bridging.
 - d. If the triangle IDNs are serial with three or more triangles, no pairing is done, and all triangles are chosen as final bridging triangles as there must be at least two connected inverted pairs in the selection.
 - e. If the triangle IDNs are not serial, there can be scattered selections (dummy IDNs [2, 3] and [7, 8] in Figure 7.20(d)). In this case, there can be one of the four possible cases occurring in pairs, in general, depending on the shape and the position of the pair of buildings (see Cases 4, 5, 6 and 7 in Table 7.8). For example, pairing between [2, 3] and [7, 8] is Ascending => [2, 3] [3, 4] [4, 5] [5, 6] [7, 8] and

Descending => [7, 8] [6, 7] [5, 6] [4, 5] [3, 4] [2, 3]. Check invert triangles in both orders and break the iteration once it meets an invert pair of triangles.

Table 7.8 Pairing algorithm of adjacent triangles: n - number of triangles and f, k, l, m, p, q, r, s and u are triangle IDNs.

Pairing of adjacent triangle IDNs (seven cases)	Case	Sorted triangle IDNs in ascending/descending order	Triangle IDN: Tidn _{i=1,n}
	1	First IDN	Ascending order => [i, i+1] _{i=1,n-1}
	2	Last IDN	Descending order => [i, i-1] _{i=n,2}
	3	Single pair [k, m]	Ascending order => [i, i+1] _{i=m,n-1} Descending order => [i, i-1] _{i=k,2}
	4	More than one pair [p,q] [r,s] [u,v]...	Consider two consecutive pairs at a time iteratively. [p,q]- [r,s] Ascending order => [i, i+1] _{i=q,r-1} Descending order => [i, i-1] _{i=p,2} [r,s] - [u,v]
	5	More than one pair with First IDN [f] [p,q] [r,s]...	Consider two consecutive pairs including first triangle at a time iteratively. [f] - [p,q] Ascending order => [i, i+1] _{i=f,p-1} Descending order => [i, i-1] _{i=p,2}
	6	More than one pair with Last IDN [p,q]... [u,v] [l]	Consider two consecutive pairs including last triangle at a time iteratively. [u,v] - [l] Ascending order => [i, i+1] _{i=u,l-1} Descending order => [i, i-1] _{i=l,2}
	7	More than one pair with both First and Last IDNs [f] [p,q]... [u,v] [l]	Similar as above in the cases 4, 5 and 6.

- f. In the example (d) in Figure 7.20, the ascending order gives out the pair [2, 3] as output and descending order gives out the pairs [7, 8] and [6, 7] as the output. Finally, combine the two outputs and remove duplicates to get the final bridging triangle selection (e.g. triangle IDNs 2, 3, 6, 7 and 8 in Figure 7.20(d)).

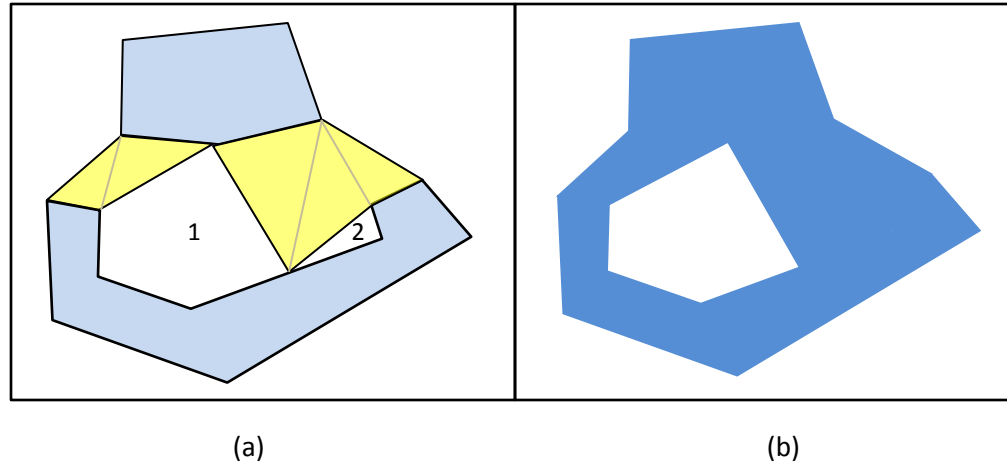


Figure 7.21 Dealing with polygon holes after aggregation of buildings with non - orthogonal sides: (a) amalgam with bridging triangles highlighted in yellow colour - rings numbered 1 and 2 are the inner holes created after bridging and (b) final amalgam after removing the triangle hole.

- g. The next step is to treat holes (inner rings) if created in the space area (Figure 7.21). In this case, first retrieve the inner holes and add any false triangles which connect to a single building if any (inner hole no. 2 in this example) to get the final amalgam (Figure 7.21(b)).
- xii. Go to step (i) iteratively until the end of all the clusters.

7.3.2 Simplification algorithm

For simplification of the shorter edges less than the simplification tolerance of the aggregated building, the algorithm implemented in the OpenCarto Java library (OpenCarto, 2013) for edge deletion is used. The principle of the algorithm is to delete all the edges shorter than a particular tolerance recursively. It starts by listing all edges shorter than the tolerance. Then, it tries to delete the shortest one. The way to delete this edge depends on the relative orientations of this edge and the two other edges connected to it (see Appendix C.3 for the algorithm). Figure 7.22 depicts the functional model of the algorithm with the filling and the simplification operations.

7.3.3 Testing of building cluster aggregation with non-orthogonal sides

In this phase, the main generalization algorithm of building aggregation with non-orthogonal sides is tested and evaluated together with the filling and the simplification algorithms. The filling operation to bridge the gaps between buildings is realised by generating triangles between buildings. Depending on the triangle edge distance threshold, the filling bridge can be enlarged in the algorithm. Further, the simplification is carried out to remove shorter sides according to the target scale.

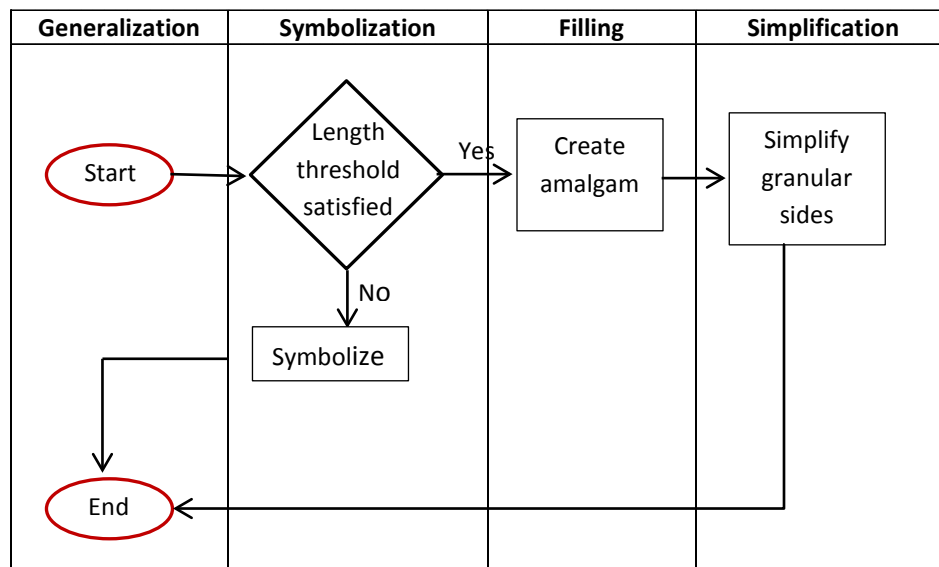


Figure 7.22 Functional process of building aggregation with non-orthogonal sides.

With a synthetic data set

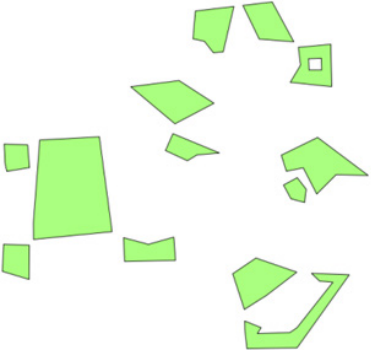
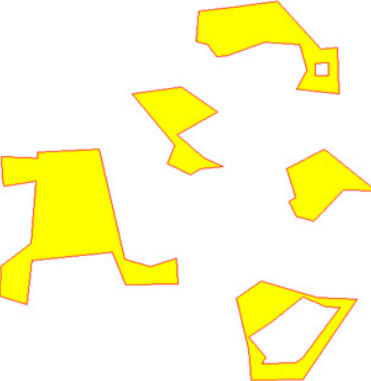
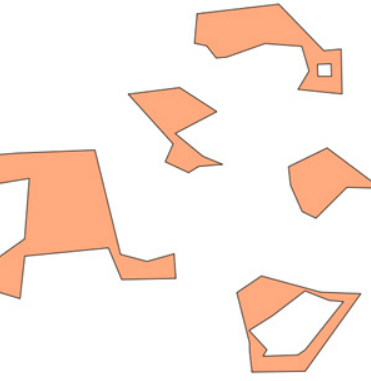
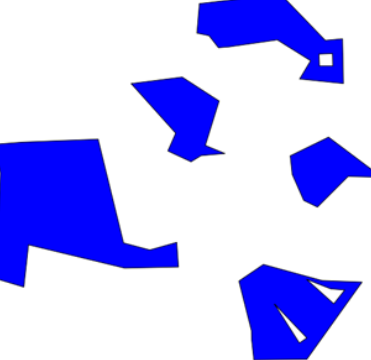
Table 7.9 illustrates the results of building cluster aggregation with non-orthogonal sides using the filling operation followed by the simplification of the shorter edges less than the simplification tolerance using an exceptional synthetic data set consisting of buildings in clusters at almost hanging positions. When comparing the results of the amalgams after aggregation with the space triangle edge distance threshold equivalent to clustering threshold (distance threshold: 20m in the example of the Table 7.9), amalgams are created with the selection of minimum number of triangles to minimise exaggeration of the shape of the amalgam (see the algorithm in Section 7.3.1). Further, the implementation of the algorithm has the capability to fill in the gaps wider between buildings in the amalgam by increasing the space triangle edge distance threshold (see the results in the example in Table 7.9 with the space triangle edge distance - 50m). According to the algorithm, if a cluster does not have buildings either touching at the corner or in overhanging positions as seen in the cluster to the mid-right in the source Figure in Table 7.9, it is amalgamated with the buffer operation, maintaining orthogonal sides if available in buildings in the cluster (see the amalgamated result in Table 7.9).

With a real data set

Depicted in Figure 7.2.3 are the clusters belonging to ‘very close’ and ‘medium range’ classifications obtained from the automatic clustering algorithm (the orientation difference threshold used is 7^0 in this example) for a region represented in a real data set. The cluster shown in red colour in Figure 7.23(a) is an orthogonal shaped cluster and therefore ignored in the process of generalization with non-orthogonal shaped clusters. It is dealt with in the generalization of orthogonal shaped clusters discussed in the previous section. Also, clusters of single buildings (buildings with cyan colour) are ignored in the generalization process.

When observing the medium range clusters - ‘MDS-1’ and ‘ML-1’ - in Figure 7.23(a), they are in almost overhanging positions and the algorithm generalizes them by bridging the gaps using triangles (Figure 7.23(b)).

Table 7.9 Results of aggregation of building clusters with non-orthogonal sides on the synthetic data with some exceptional building configuration.

Geometry	Distance threshold in meters (m)	Visual representation
Source building data in the form of clusters	Clustering - 20	
Building amalgams after aggregation**	Space triangle edge threshold - 20 (same as clustering threshold)	
Building amalgams after aggregation* and simplification*	i. Space triangle edge threshold - 20 (same as clustering threshold) ii. Simplification tolerance - 10	
Building amalgams after aggregation* with filling** and simplification*	i. Space triangle edge threshold - 50 ii. Simplification tolerance - 10	

* Existing algorithm

** New algorithm

At the bottom of Figure 7.23(b) is the generalized result of the complex cluster - 'VC-3' - comprising of irregular shaped and differently oriented buildings at both overhanging and non-overhanging positions. The simplified results using the open source OpenCarto Java library are shown in Figure 7.23 (c) and (d). The cluster with 'ML-2' classification is symbolized, considering the building length threshold. However, if it is required to have a little more generalized aggregation for complex clusters and/or large clusters such as the cluster - 'VC-3' - the improved concave hull algorithm described in Section 5.4.1 can be used.

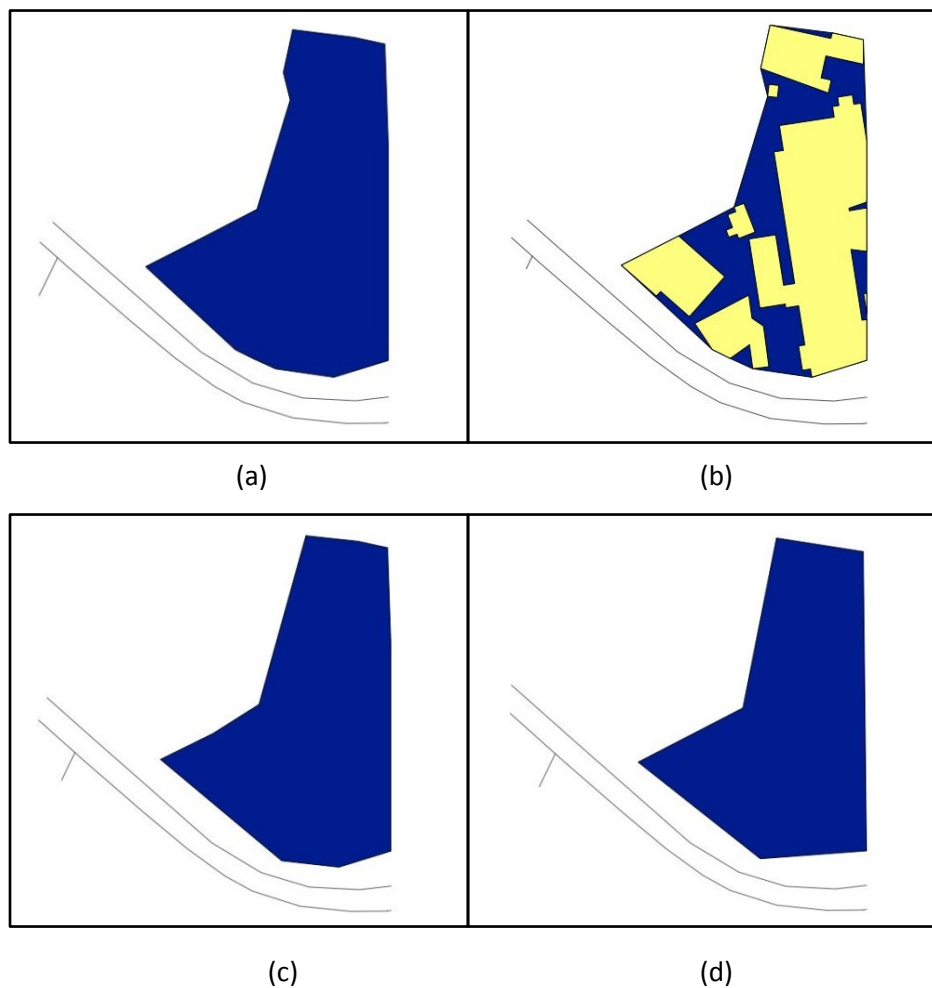


Figure 7.24 Results of the generalized amalgams with the improved concave hull algorithm: (a) amalgam created with concave hull generation (b) amalgam overlaid with the building cluster (c) simplified amalgam with the OpenCarto Java library and (d) simplified amalgam with the JTS Java library. Simplification threshold in both cases - 10m.

Figure 7.24 depicts the results of the generalized amalgam of the aforesaid cluster - 'VC-3' - using the improved concave hull algorithm. When comparing the result with the one produced by the triangulation based aggregation algorithm, it is overgeneralized with some improved results. Results (c) and (d) in Figure 7.24 depict the two simplified results generated by the two simplification algorithms implemented in the OpenCarto (see Appendix C.3 for OpenCarto algorithm) and the JTS Java based libraries using the same distance threshold. The algorithm used in JTS is similar to that of Douglas and Peucker (1973) with further improvement to have a result with valid geometry with the preservation of topology (Vivid Solutions JTS, 2013). When comparing the results, the JTS simplifier has given more simplified results that would suit simplification of polygons generated by the improved concave hull algorithm on larger amalgams.

Synopsis of the contributing algorithms used

Table 7.10 summarises the contributing algorithms of the main building aggregation algorithm with non-orthogonal sides.

Table 7.10 Building cluster aggregation algorithm with non-orthogonal sides, including contributing algorithms.

Main algorithm	#	Contributing algorithm	Purpose
Building cluster aggregation with non-orthogonal sides	1	Dilation and erosion with the buffering algorithm I [*]	To fill the gaps between building geometries
	2	DCT ^{**}	To get the adjacency relations between building geometries
	3	CNDT ^{**}	To fill the gaps between buildings with triangles
	4	Edge adjacency graph [*]	To sort the adjacent triangles
	5	Filling with the pairing of the adjacent triangles ^{***}	To fill the gaps between buildings with the candidate triangles
	6	Concave Hull ^{**}	To aggregate larger clusters
	7	Simplification [*]	To delete shorter edges

^{*} Existing algorithm

^{**} Modification and/or extension to an existing algorithm

^{***} New algorithm

7.4 Conclusion

The chapter describes how the existing algorithms are tested and the new algorithms are developed for the purpose of creating generalized building polygons to depict salient landmarks on a coarse background to generate focus maps for wayfinding. For creating generalized amalgams, building cluster information enhanced under the data enrichment described in Chapter 5 is used. Further, in the development and the implementation process of each algorithm, a thorough testing has been carried out by analysing the results of its implementation on several versions until the refined algorithm is obtained using both the real and the synthetic data sets. The amalgamation algorithms developed in this work fall into two categories: (a) algorithm to deal with buildings with orthogonal sides and similar orientation in the cluster outline and (b) algorithm to deal with clusters comprising of buildings with non-orthogonal sides and/or dissimilar orientations in the cluster outline.

The next chapter will describe how the tools and methods in the field of spatial data structures, data enrichment, data mining and automatic map generalization, implemented in this research are utilised together to generate focus maps for wayfinding along with the external validation of the results of focus maps.

Chapter 8 Results and discussion

This chapter elaborates generating focus maps, together with external validation of its results in the field of automatic map generalization and data mining for deriving landmark saliency. The generalized results are validated with the use of proprietary ArcGIS software as described in Section 3.3.3. The derived landmark saliency is validated through the implementation of the framework to find landmark saliency, introduced by Raubal and Winter (2002) and its extended framework by Nothegger, Winter and Raubal (2004) as described in Section 3.3.4 (see Appendix F.6 for the UI of the implementation of both frameworks). The validation of landmark saliency is further analysed with a crosscheck by using the Google street view of the test areas. This chapter further describes the implications and limitations of the results of focus map generation in relation to the literature review given in Chapter 2. Two different data sets - one from the London Borough of Newham and the other from the London Borough of Tower Hamlets - using OS MasterMap data at the scale of 1 : 1.25K are used for this purpose. The final product of focus map generation is achieved by the integration of results obtained through the implementation techniques described in Chapters 4, 5, 6 and 7 in the areas of spatial data structuring with triangulation, data enrichment in the context of automatic map generalization, data mining and automatic map generalization. In this process, the results of each stage are thoroughly analysed since the results of one phase affect the next in generating focus maps.

8.1 Dealing with issues in generating the results for focus maps

In producing focus maps, the results at each stage, as mentioned above, can have a significant effect on the next stage towards generating the final results. When applying to the real data (as opposed to the synthetic test data), a few immediate fixes were required in the phases of data enrichment and automatic map generalization in this research. Therefore, fixing will be dealt with next before producing the focus maps on the test data sets.

8.1.1 Fixing issues in building clustering

In the building clustering process under data enrichment phase, the execution of the CNDT with building edges set as constraints failed in some of the regions in the OS MasterMap data sets. However, when the tolerance was increased to merge site points closer than a certain distance to deal with robustness issues, the triangulation executed, but did not give topologically correct triangles (no triangle edges existed from some site points that shared attached building edges and some triangle edges went across constrained geometries (Figure 8.1(a)), resulting in incorrect clustering.

The ideal application would have been to run triangulation with a tolerance of zero or a very small value (0.1mm). However, for data in some regions, the tolerance had to be increased interactively until the triangulation executed despite incorrect topological results. Thus, an immediate fix was necessary. Identifying that the issue was partly in the data set with incorrect noding issues and duplicate geometries, cleaning and building topology with the GIS software - GRASS - was carried out before applying triangulation. However, this strategy too failed. Then the linear geometries set as edge constraints in the data set were tested and found that there existed duplicate line segments. The duplicate line segments had to be dissolved so as to get a unique constraint segment list to be input into the triangulation. This strategy worked out and with the modified triangulation method, along with a very small tolerance set in the code (0.1mm), topologically correct triangles could be generated with full automation (Figure 8.1(b)).

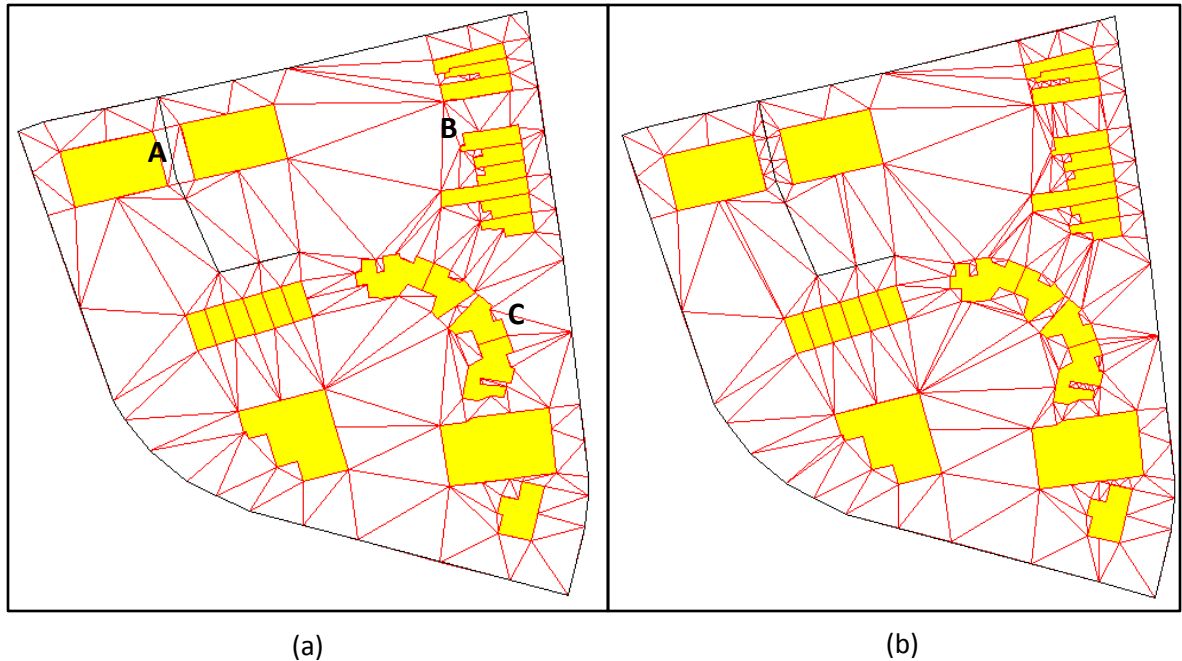


Figure 8.1 Results of the CNDT with the edge constraints: (a) topologically incorrect triangulation: **A** - a triangle edge runs across the road, and **B** and **C** - non-triangulated building corner points and (b) topologically correct results after the fix. Data source: OS MasterMap data of Newham area at the scale of 1 : 1.25K.

Figure 8.2 depicts the clustering results of buildings in a region before and after the fix of the duplicate geometries in the data set. Although the road network is used as a constraint in clustering, due to the increase of snapping tolerance in the triangulation, buildings highlighted in yellow colour in Figure 8.2(a) have been considered as a topologically adjacent pair in the triangulation. As a result, they have been grouped into a single cluster with a similar shape in the medium distance range. After treating duplicate geometries with triangulation, this pair is split into single buildings using the constrained road segment as shown in Figure 8.2 (b).

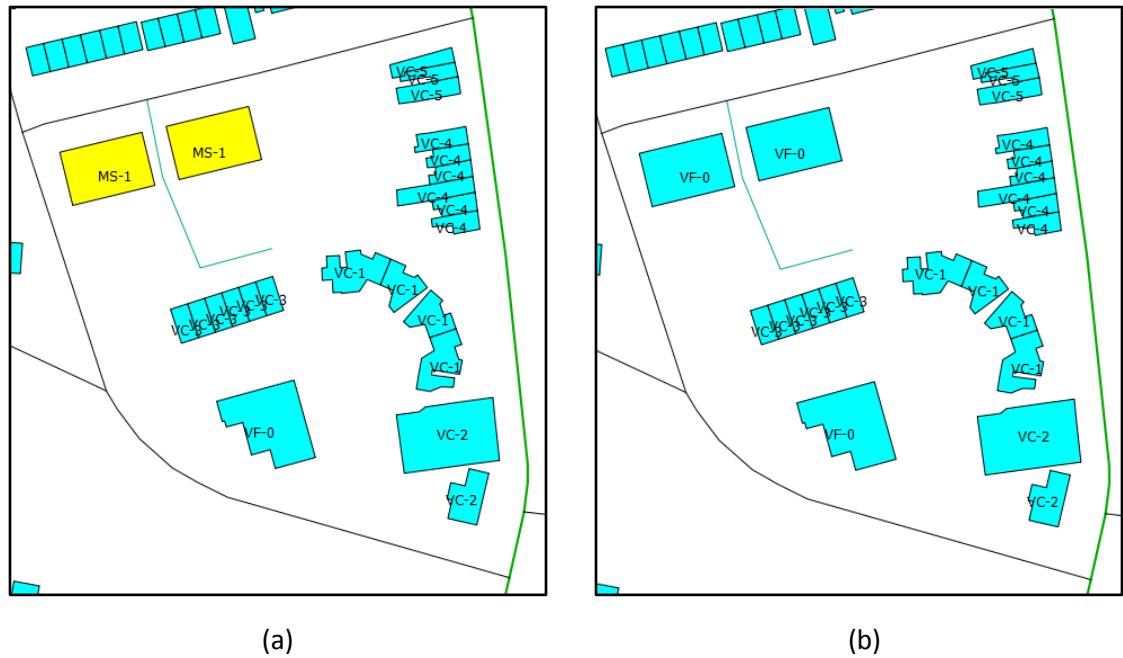


Figure 8.2 Results of clustering: (a) incorrect clustering due to topologically incorrect triangulation (highlighted in yellow colour) and (b) correct clustering after the fix of CNDT with the edge constraints. Clustering distance thresholds: Very close - 0.5mm, medium - 2mm and very far - 2mm of the target map scale of 1 : 5K. Data source: OS MasterMap data of Newham area (part of) at the scale of 1 : 1.25K.

8.1.2 Fixing issues of enrichment of building information for deriving salient landmarks

When analysing the automatic results of the orientation of each building to the road network with the use of building to road adjacency relationships derived from the CNDT, it was found that the implementation of the algorithm described in Section 6.1.1 collapsed due to a null pointer reference exception. The reason was that when the angle of the road segment formed between the two intersection points derived using the extended parallel line segments on either side of the longest side of the LOBR of a building under data enrichment (see Figure 6.4, page 174), there were instances where one of the two lines did not intersect the road (see building highlighted in yellow colour with the attribute 'orientation to road' classified as 'none' in Figure 8.3(a)). In these instances, the angle of the road segment became null causing the null pointer exception. The other problem found was that the road segment thus formed was too simplified from the original road

segments when there was a sharp turn on the road, causing incorrect interpretation of the angle between a building and the road, assigning a wrong orientation (see Figure 8.3(b)).

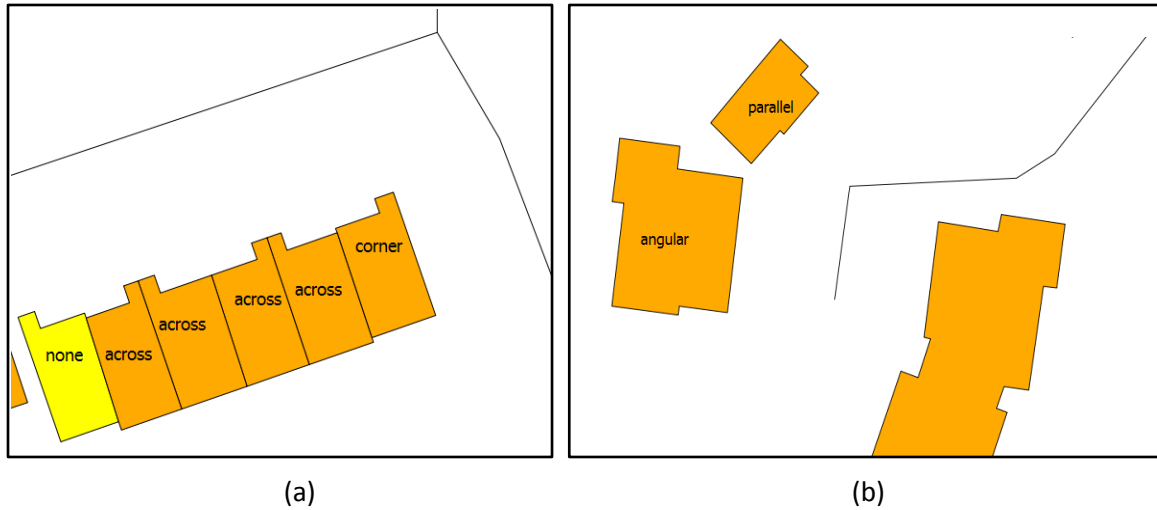


Figure 8.3 Results of building orientation to the closest road: (a) building highlighted in yellow colour is not assigned an orientation due to the exception explained above and (b) incorrect assignment of the orientation to the road for the two buildings to the left of the road. Data source: OS MasterMap data of Newham area (part of) at the scale of 1 : 1.25K.

The exception was fixed by modifying the step to find the line segment of the road using the improved algorithm described in Section 6.1.1 with the following steps.

- When the line from the centroid of the LOBR of the building to the midpoint of one of its perpendicular sides is extended to intersect the road (see line from points **c** to **mp₂₃** in Figure 6.4, page 174), get the intersection point and retrieve the road line segment that lies at the intersection point.
- Find the angle of that particular road segment to be used in the calculation of finding angle difference between a building and the road in the algorithm.

Figure 8.4(a) shows the orientation of the building highlighted in yellow colour to the road as 'across' after fixing the exception while Figure 8.4(b) shows the correct assignment of the orientation of the two buildings to the left of the road.

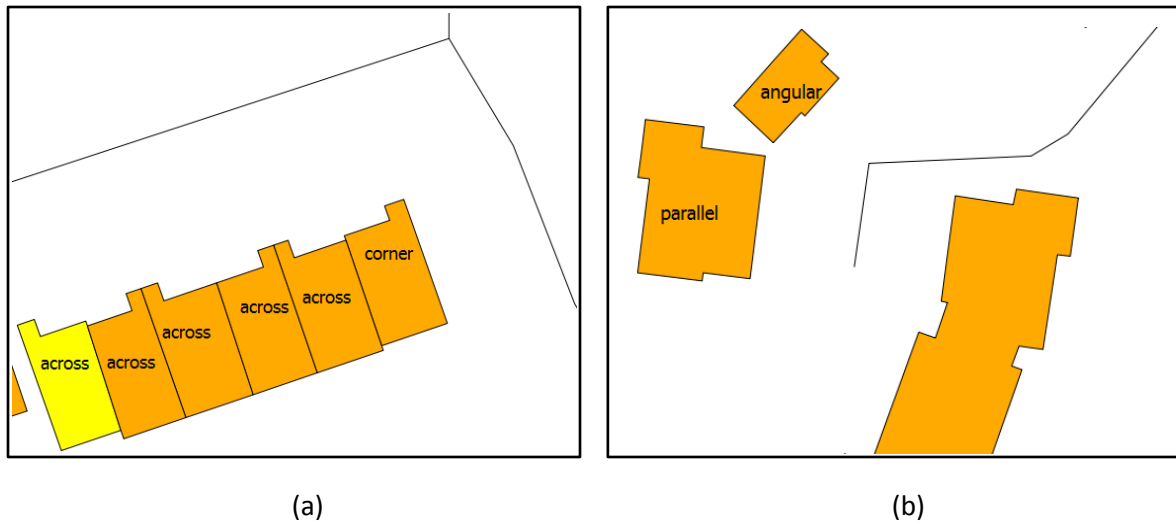


Figure 8.4 Results of building orientation to the closest road after fixing the exceptions: (a) highlighted building is assigned the correct orientation and (b) correct assignment of the orientation to the road in the two buildings left to the road. Data source: OS MasterMap data of Newham area (part of) at the scale of 1 : 1.25K.

In this algorithm, in order to enrich the orientation of buildings close to road corners, two or more intersecting roads at a node (junction of turning point), linked to a building are considered using the building to road adjacency relations derived from triangulation. However, this approach does not always produce expected results, depending on the configuration of buildings and road network (e.g. in the case of adjacent buildings in a row, which share a common edge, located near a junction, the buildings little away from the junction would also be assigned as ‘corner’ since they could be linked to two or more roads (see Figure 8.5(a)). One of the difficulties to get a unique result was that depending on the distance from a road intersection to buildings, the classification of a building as a ‘corner’ varied. One of the solutions was to densify the road network before running the CNDT to get more hooks to restrict a building linking with many roads in the network. However, this approach too was only a partial solution to the issue (see Figure 8.5(b)) after densification of the road network where only two incorrectly classified buildings were rectified. Such unexpected results in a relatively few cases overall had to be interactively rectified before proceeding into emphasising salient landmarks under the data enrichment phase.



Figure 8.5 Results of building orientation to the closest road, located at corners: (a) buildings highlighted in yellow colour with incorrect orientation and (b) orientation to the road of the same set of buildings after densification of the road network at 3m intervals where highlighted buildings have incorrect orientations to road. Data source: OS MasterMap data of Newham area (part of) at the scale of 1 : 1.25K.

8.1.3 Fixing issues in the aggregation algorithms developed

Aggregation algorithm with orthogonal sides

It was identified that the enlargement of juts did not work out correctly when observing the result of aggregation of the pair of buildings in Figure 8.6 (a). This issue occurred in a few clusters only.

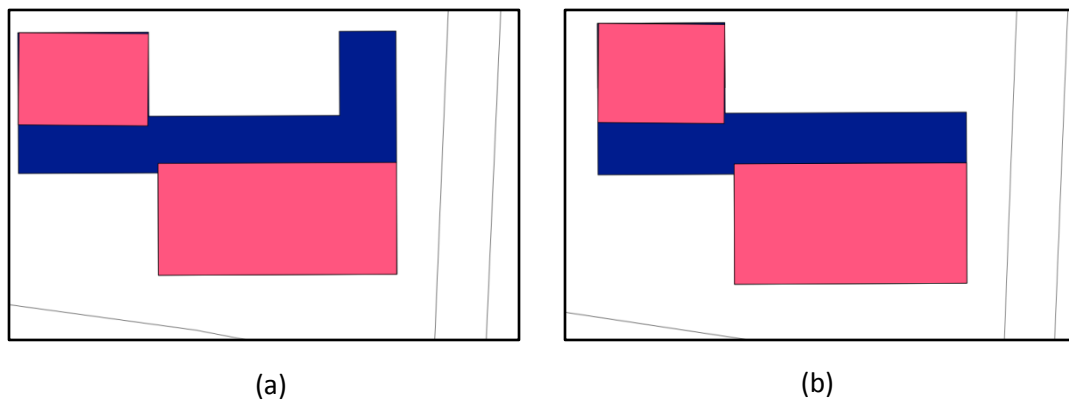


Figure 8.6 Enlargement results after squaring with a distance 5m: (a) faulty enlargement with an additional strip standing vertical and (b) correct enlargement along the jut between the two buildings.

When investigated, the cause was in the derivation of outer space polygons in step (vi) in the enlargement algorithm I and used in the enlargement algorithm II as described in Section 7.2.5, page 210. In deriving space polygons, the algorithm cut out the squared amalgam from its MBB, using the polygon difference method. When this was done, due to the robustness issues caused by limitations in the use of finite precision numerical values in overlay computations in geometric algorithms (polygon difference algorithm), the resultant space polygons sometimes did not create the exact outline near the boundary between the amalgam and the MBB. As a result, in checking the space polygons that shared an edge with the inner polygon strips along the X and the Y directions in the enlargement algorithm (see steps (x) and (xi) in Section 7.2.5, page 212 of the enlargement algorithm I, used in the algorithm II), some unwanted space polygons were selected. Thus, the algorithm assumed that the particular strip was to be enlarged from the selected sides of such unwanted space polygons. In order to fix the problem (though not a problem in the algorithm, but an implementation issue), steps (ix) to (xiv) in the enlargement algorithm I used in the algorithm II was replaced with the following steps to deal with this issue (see Appendix G.6 for the pseudo code):

- i. Iterate through slices in each X_strip stack separately and check if each slice has got space polygons on either side or single side using a 'point in polygon test', creating two arbitrary points on either side of the slice (left and right) with a small threshold distance.
- ii. Iterate through slices in each Y_strip stack separately and check if each slice has got space polygons on either side or single side using a 'point in polygon test', creating two arbitrary points on either side of the slice (top and bottom) with a small threshold distance.
- iii. If a slice has got space polygons on both sides (either left and right or top and bottom), union the slices of the respective strip and then generate two even rectangular polygons as enlargement candidates on either side using the MBB of the union strip together with the union strip width and the enlargement width (Figure 7.9, page 213).

- iv. If all slices of the strip have got space polygons only on one side (top or bottom in the case of Y_strips and left or right in the case of X_strips), union the slices and check if the opposite side of the union strip adjoins the MBB of the amalgam. If it adjoins, fill the other side of the strip using its MBB together with its width and the enlargement width (Figure 7.9(d), page 213). If it does not adjoin the MBB of the amalgam, ignore the particular strip.

Figure 8.6(b) shows the enlargement result after revising the enlargement algorithm to fix the robustness issue in the implementation (see Appendix G.8 for the pseudo code of the aggregation algorithm with orthogonal sides).

Aggregation algorithm with non-orthogonal sides

The first issue identified was treating the self-connecting bridges during the buffer operation in the case of concave corners and holes in the source building polygons as explained in Section 7.3.1, page 230. When applied the dilation and erosion technique to buffer operation using the same distance, it changed the original vertices in the outer ring of the polygons subjected to the buffer in the cluster due to geometry precision exceptions. Therefore, when subtracting the buffered result from the union geometry of the cluster, sliver polygons occurred in some instances along the outer ring of the buffered result, causing implementation to bring a runtime exception. This operation of dealing with self-connecting bridges was omitted from the algorithm since it could seldom happen in exceptional building geometries and would not bring any adverse effect on the final results. However, the incorporation of building holes was necessary and treated after obtaining the amalgam. The aggregation algorithm was modified so that it could either preserve or remove inner holes based on an area threshold during the aggregation operation.

The next major issue was aggregating bridges between each pair of building polygons with triangles formed using the CNDT with edge constraints. Since some vertices were not part of the original polygon because of the addition of Steiner points to make the triangulation Delaunay stable, merging triangles to fill the bridges would sometimes create multi-

polygons despite merging but seemed to be visually satisfactory. However, this did not create any runtime exception in the implementation. The intention of using the CNDT to fill the gap between each pair of buildings was to have bridges with the minimum exaggeration as explained in Section 7.3.1, page 233. Therefore, the DCT algorithm developed in this research was used to fill the gap between each pair of buildings instead of the CNDT. In order to have the least exaggeration of filling bridges, the densification of polygons was performed first with a distance equal to the cluster distance before applying the DCT algorithm (see Appendix G.9 for the pseudo code of the aggregation algorithm with non-orthogonal sides).

8.1.4 Fixing issues in the simplification of aggregated amalgams

The OpenCarto simplification algorithm described in Section 7.3.2, page 238 gave exceptions in edge simplification even on squared amalgams after aggregation (Figure 8.7(b)). When investigating the algorithm, it was found that it did not handle intermediate vertices of building edges if existing in the input building geometry. The reason was that the algorithm considered two consecutive vertices as an edge. Therefore, if building edges consisted of intermediate vertices of a building geometry, they were removed before the application of the algorithm. Thus, the simplification of intermediate vertices used in the improved concave hull based algorithm described in Section 5.4 will be used with a small angle tolerance, terminating the removal of intermediate vertices when the building polygon has got only four edges in the vertex simplification process.

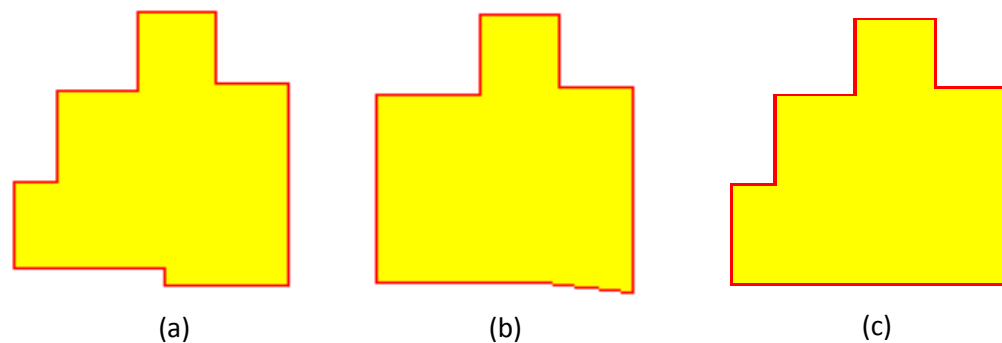


Figure 8.7 Simplification of a building with the OpenCarto edge deletion algorithm: (a) source building (b) target building after simplification where the orthogonality does not preserve and (c) simplification with squared edges after removal of the intermediate vertices that existed on the edge at the lower left corner of the source building.

8.1.5 Further incorporation and development of generalization tools required for deriving focus maps

Polygon fusion

When investigating the OS test data sets, it was found that there were a number of buildings adjoining to each other sharing a common edge (attached buildings). In these cases, fusing of buildings in clusters (the term fusion is according to the definition of generalization operations defined in the AGENT project (Figure 2.5, page 21)) was done with the use of the PostGIS ST_UnaryUnion method without amalgamating (merging) building clusters. The ST_UnaryUnion with ST_Collect method has the ability to dissolve boundaries of geometries in a geometry collection (see Appendix E.1 for a sample SQL used). However, there were cases with a mix of attached buildings and isolated buildings in a cluster. In these cases, the attached buildings were fused with the same functions during the PostGIS query before applying one of the two aggregation algorithms developed in this research, depending on the cluster shape (see specification for automatic generalization in Section 8.2.1).

8.2 Results of focus maps

It is important to note that the focus maps derived in this research are not necessarily a cartographically aesthetic product with various cartographic generalization techniques applied to depict landmark saliency as already researched by Sester (2002) and cited by Elias, Hampe and Sester (2005). Therefore, the salient landmarks derived are portrayed in the original form on the coarse background created by the automatic map generalization on the focus map. More emphasis has been placed on investigating the gaps that need to be filled during the process of focus map generation with the least graphical conflicts (distortions) to maintain quality information in the generalized output. This quality information on a coarse background is important to keep users engaged in a wayfinding task. The model of generalization used in this research is more towards the statistical generalization operations such as selection, removal and aggregation (aggregation is the main operation dealt with in this research) described in Section 2.2.3, page 18, although

the cartographic generalization operations such as simplification, enlargement and exaggeration are used together with the aggregation generalization operation, ensuring no graphical conflicts at the target scale. The digital products in the test areas thus generated can be used to create a scenario for adaptive visualisation of landmarks using the MRDB concept adopted by Elias, Hampe and Sester (2005) as described in Section 2.2.7 in a real-time navigation system where the notion is to present the user with a coarse background of building details (e.g. generalized coarse building information in this research), allowing the user to visualise salient landmarks dynamically on their original shape as he/she approaches a salient landmark on the route or at a decision point, removing the coarse background amalgam pertaining to a particular landmark. The salient landmarks can be more emphasised by using a graphical variable - colour - as discussed by Reichenbacher (2004). Linkage between the salient landmark and the amalgam with 1 : N relationship can be established during the generalization process as mentioned in Section 2.2.7 and stored in the spatial database for removal of the amalgam in order to emphasise salient landmarks.

Each of the processes in deriving salient landmarks was tested using data both from Sri Lanka and the United Kingdom. However, the final results are presented with the data only from the United Kingdom due to the inadequacy of building height information and non-availability of the Google street view of data from Sri Lanka for deriving landmark saliency and validating of the same respectively.

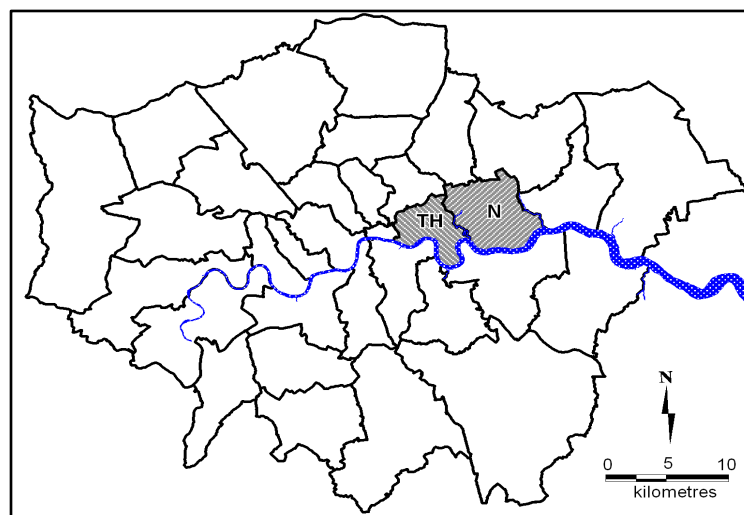


Figure 8.8 Location map of the London Boroughs of Newham and Tower Hamlets (hatched).

Further, all the algorithms developed and/or modified in this research have been implemented to process data within a defined region. The advantage of processing data by region is that it reduces the complexity of handling a large volume of data as well as improving runtime efficiency of handling complex algorithms dealing with spatial geometries. Figure 8.8 depicts the location map of the two test areas used to generate focus maps.

Newham test data

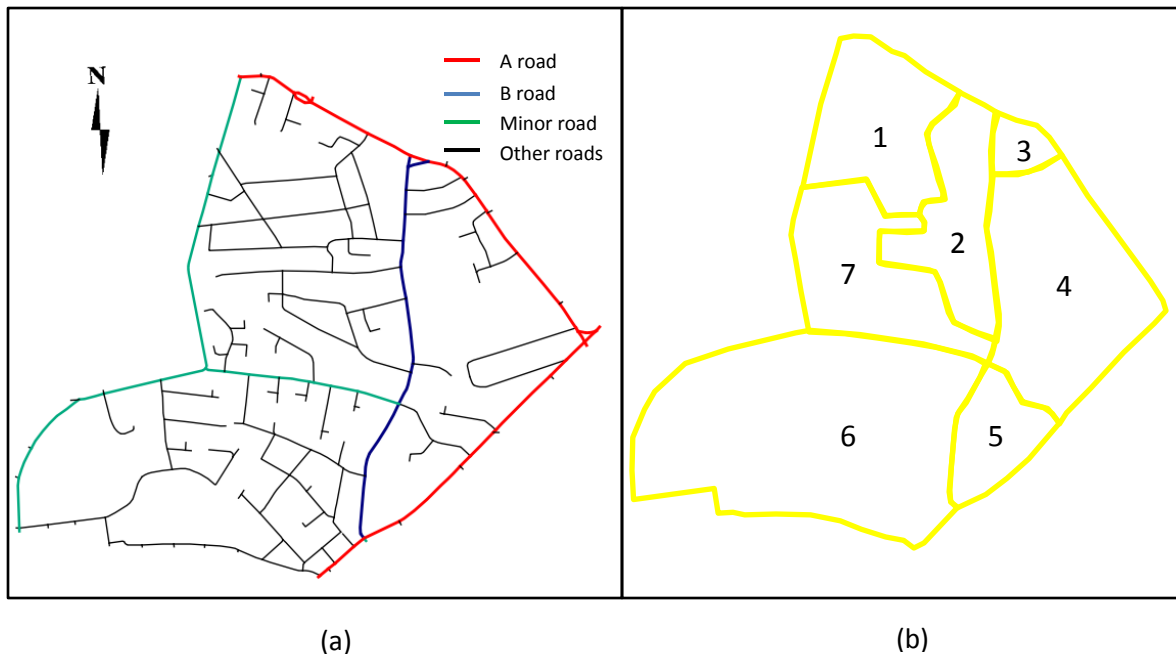


Figure 8.9 Newham test area: (a) road network with the classification of roads comprising of minor roads, private roads and alleys and (b) chosen regions (area partitions) for data processing during the focus map generation.

The test data from Newham area were chosen with a mix of building geometries comprising in regions interactively partitioned as shown in Figure 8.9(b) for the generation of the focus map. Since the modified clustering algorithm described in Section 5.3.3, page 148 can take into account contextual features, it is possible to partition the data set into meaningful chunks as desired with the help of contextual geometries. Thus, a region can be defined with several small sub-regions inside. This is very useful when there are many small regions surrounded by the contextual features as characterised by the data sets in the test areas of Newham and Tower Hamlets where there are small regions surrounded

by local streets, private roads and alleys. Therefore, a few adjoining smaller regions in these data sets were merged into a meaningful single region, considering building configuration before the start of focus map generation (Figures 8.9 and 8.13). Ultimately, the data were processed region-wise during the generation of the focus maps. Figure 8.10 depicts the source map of Newham area at the scale of 1 : 8K. The coarse background building information generalized at a target scale of 1 : 8K mainly with building aggregation for the focus map generation is depicted in Figure 8.11 while Figure 8.12 depicts the focus map with salient landmarks highlighted with the graphical variable - colour - on the coarse background.

Source map of Newham area

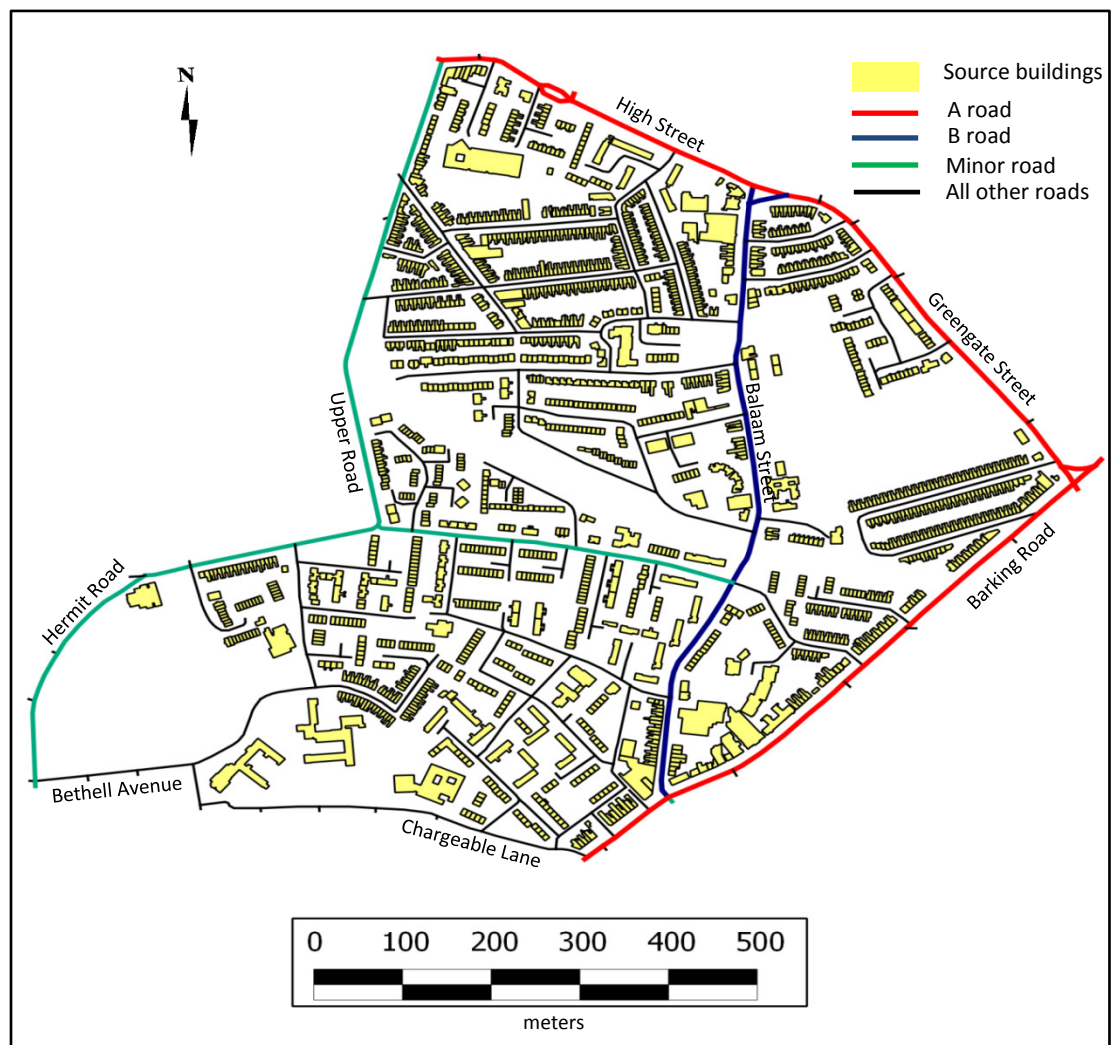


Figure 8.10 Source map of Newham test area at the scale of 1 : 8K. Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

Generalized background of the focus map – Newham area

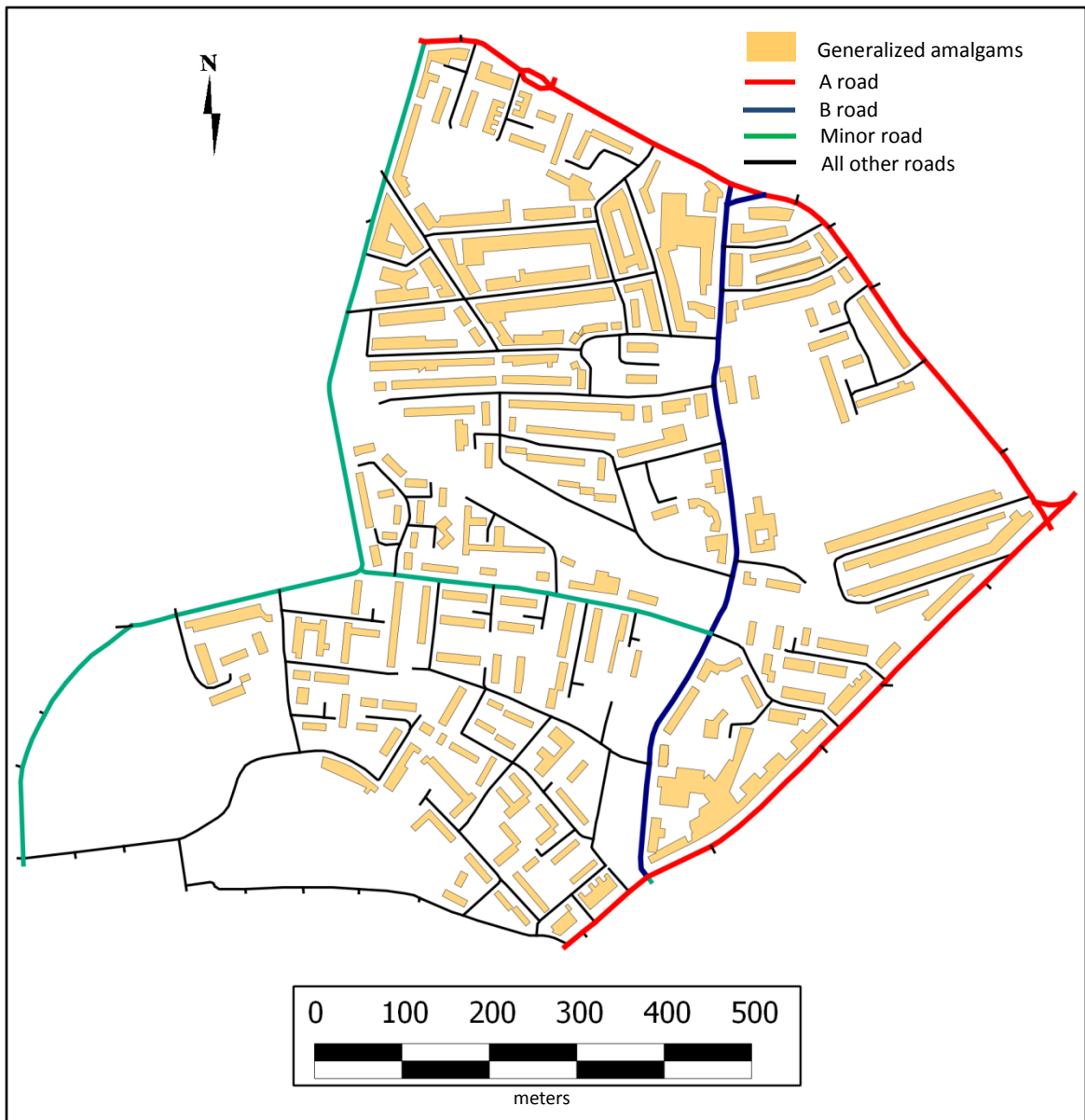


Figure 8.11 Generalized building amalgams at the target scale of 1 : 8K to be used as the background on the focus map, derived from the source data at the scale of 1 : 1.25K in Newham area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

The focus map – Newham area

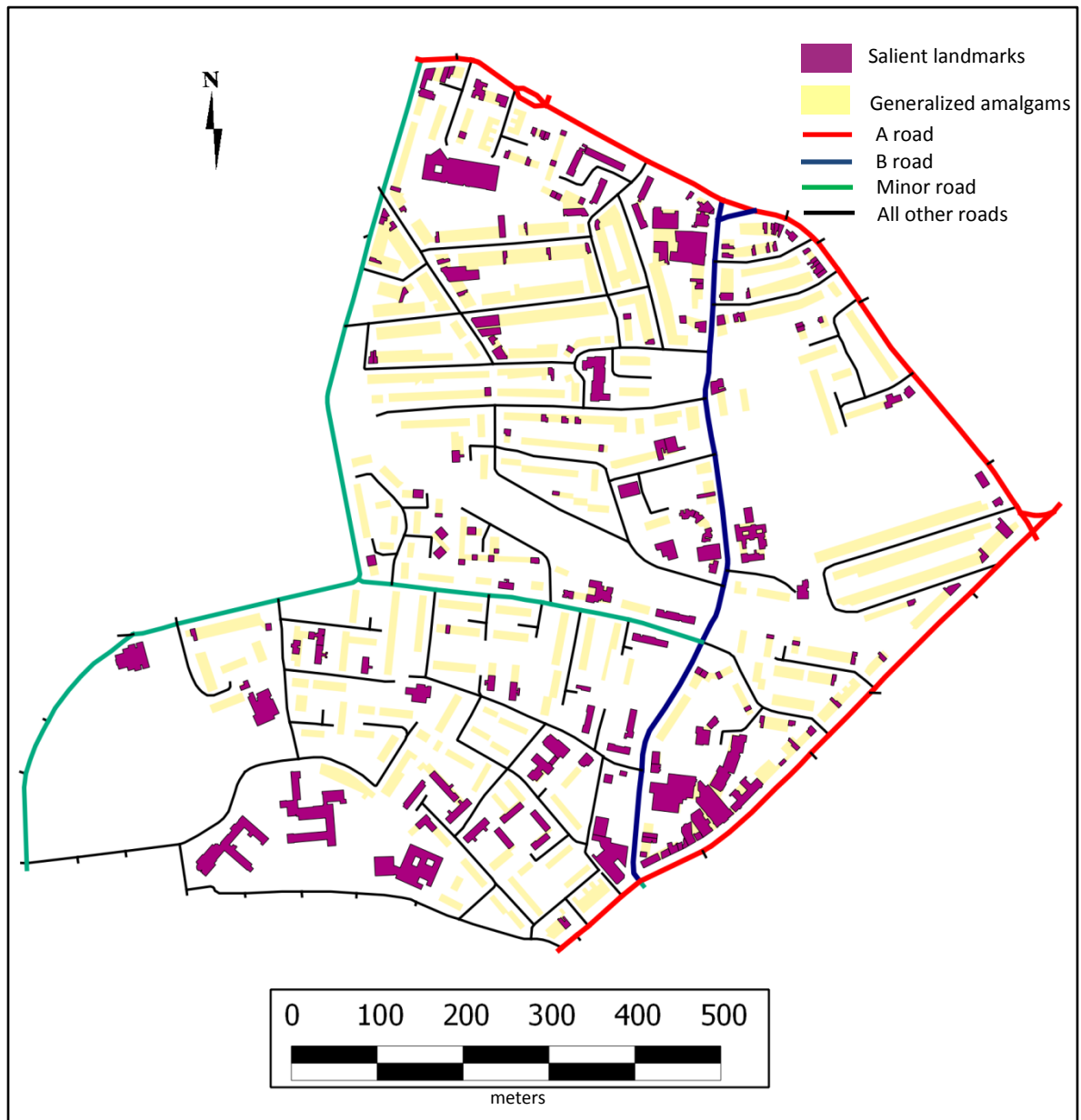


Figure 8.12 Focus map with salient building landmarks highlighted with the graphical variable - colour - portrayed in the original shape on the coarse background of the generalized buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Newham area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

The scale 1 : 8K is selected for depicting the coarse background since it is a medium scale which does not apply excessive generalization on data as in the smaller scales such as 1 : 25K and 1 : 50K. A highly coarse background is especially not suitable for maps designed for pedestrian wayfinding where the user should have the opportunity to

understand the context in the surroundings while walking. Small scale coarse backgrounds on focus maps are especially suitable for users who travel by vehicles.

Tower Hamlets test data



Figure 8.13 Tower Hamlets test area: (a) road network with the classification of roads comprising of minor roads, private roads and alleys and (b) chosen regions (area partitions) for data processing during the focus map generation.

The test data from Tower Hamlets area consist of some very complex building geometries including structures. The test area was chosen to represent complex high-rise buildings and structures with a variety of different and similar heights. Regions were partitioned as shown in Figure 8.13(b) for generating the focus map. Figure 8.14 depicts the source map of Tower Hamlets area at the scale of 1 : 8K. The coarse background building information generalized at a target scale of 1 : 8K for the focus map generation is depicted in Figure 8.15 while Figure 8.16 depicts the focus map with salient landmarks highlighted with the graphical variable - colour.

Source map of Tower Hamlets area

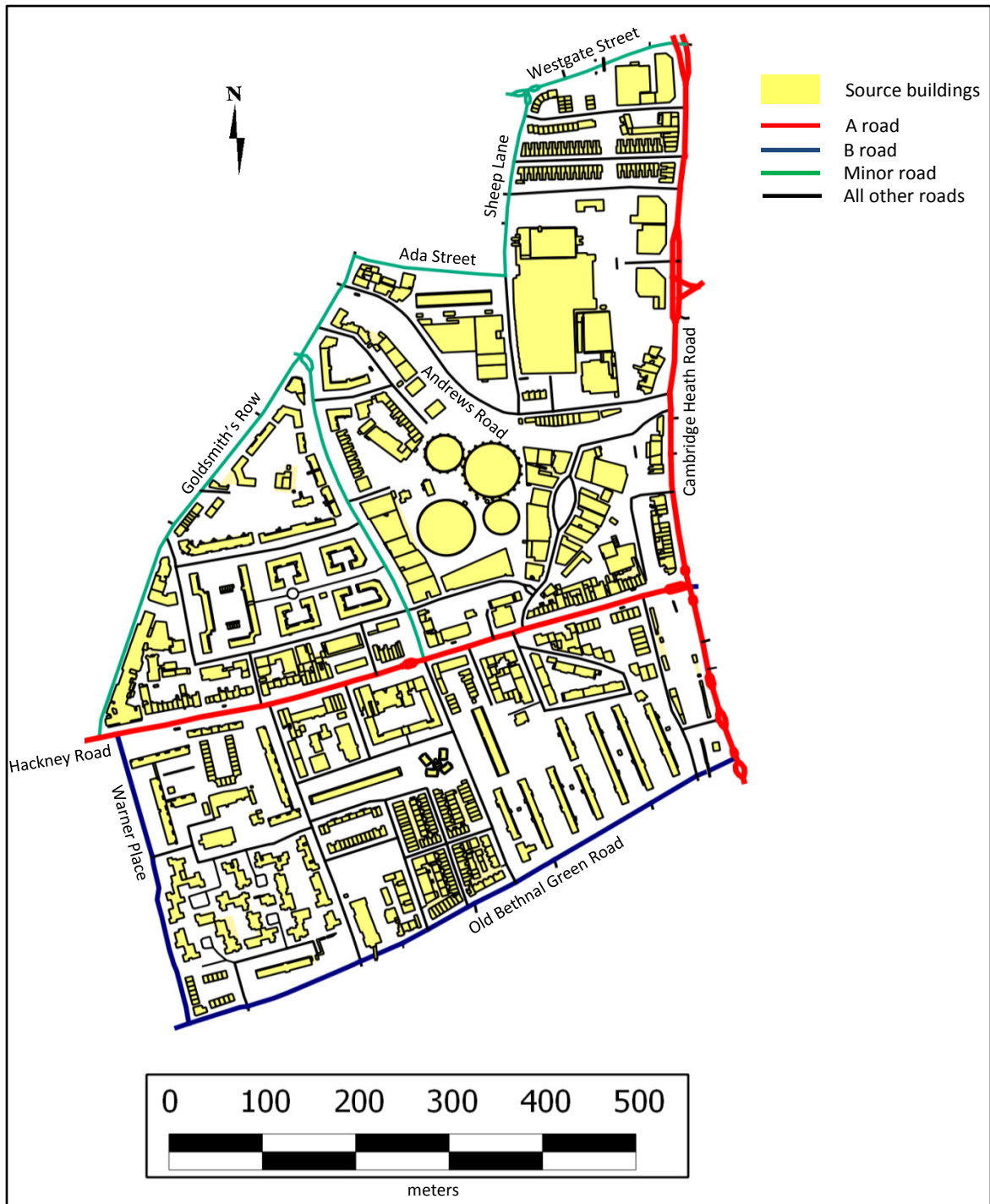


Figure 8.14 Source map of Tower Hamlets test area at the scale of 1 : 8K. Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

Generalized background of the focus map – Tower Hamlets area

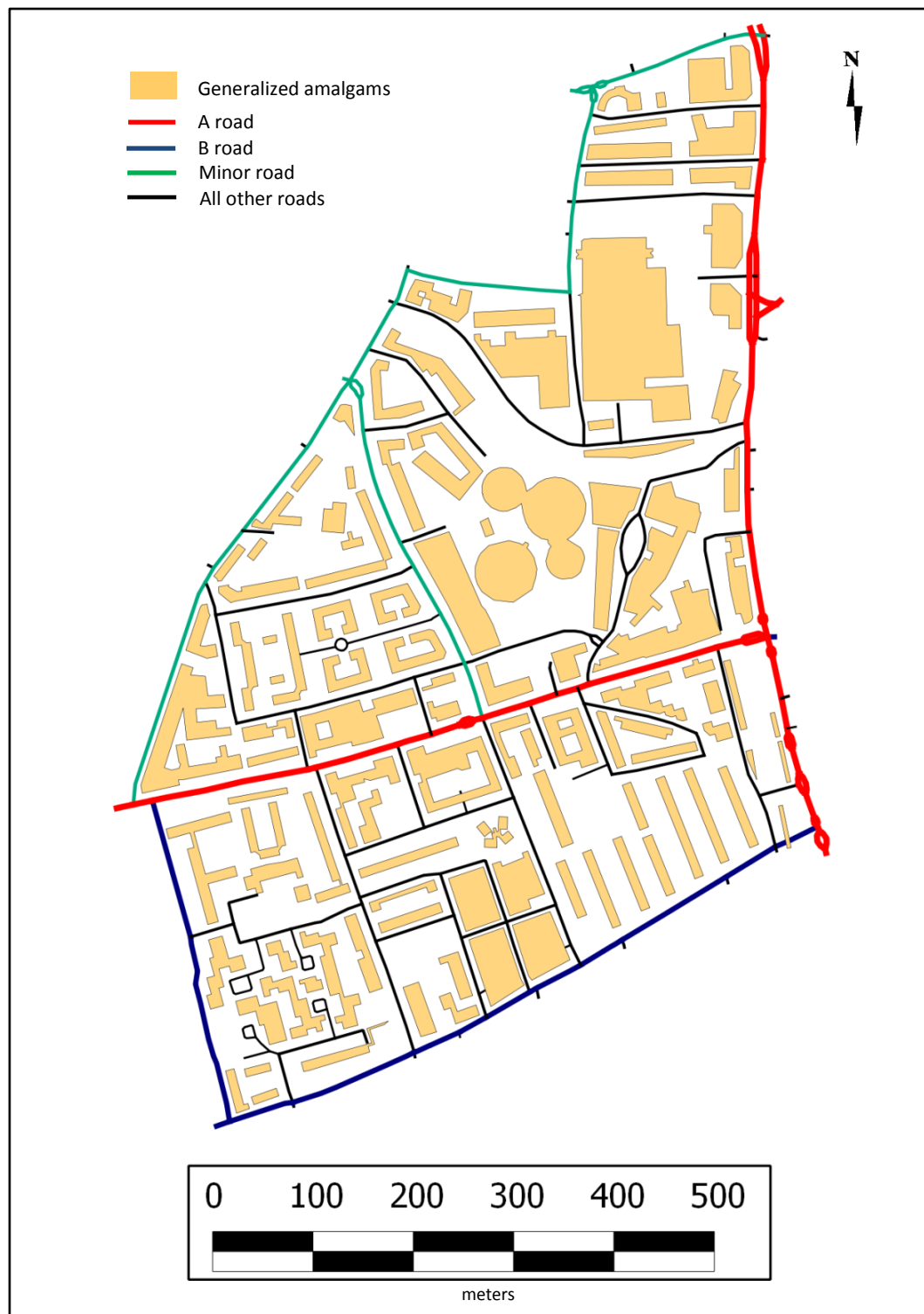


Figure 8.15 Generalized building amalgams at the target scale of 1 : 8K to be used as background on the focus map, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

The focus map – Tower Hamlets area

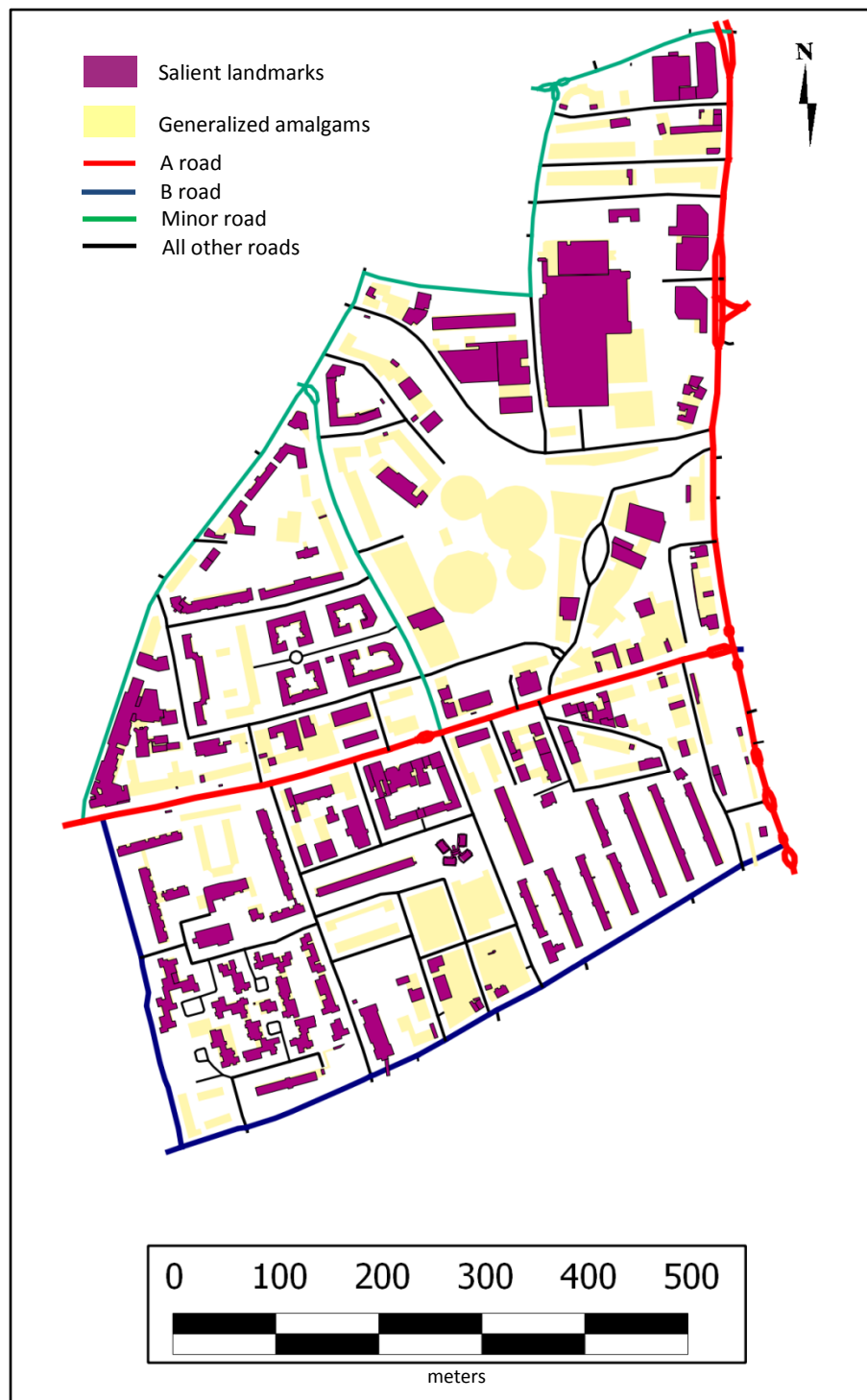


Figure 8.16 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized amalgams of buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

8.2.1 Specifications used in the focus map generation process

Data enrichment process for the automatic map generalization

Clustering distance threshold values:

- i. The distance threshold values used for the evaluation of clustering in the Phase I, described in Section 5.3, page 126, are used.
- ii. The same similarity index (0.75) used for the evaluation of clustering in the Phase I is used.
- iii. The orientation threshold value used in the clustering evaluation in the Phase I was 7° . However, there was a necessity in Phase II to revise the threshold value according to the results of the cluster evaluation given in Section 5.3.3, page 145, as it was difficult to identify the orientation difference between a pair of buildings smaller in size with the value of 7° . Therefore, the orientation threshold value used in the generation of clusters in the focus map generation is 15° .

Cluster shape enrichment:

- i. Clusters are considered to be non-orthogonal if the maximum orientation difference between buildings is greater than 15° and/or if at least a single building in the cluster is non-orthogonal.

Data enrichment process for deriving salient landmarks

- i. A building is considered to be angular if the longest axis of the LOBR of a building is inclined to the closest road segment with an angle, not within a region of $90^{\circ} \pm 15^{\circ}$.

Automatic generalization process for deriving coarse background on the focus map

Automatic generation of the coarse background with generalized amalgams in this research adheres to a set of generalization specifications in the application of the

aggregation algorithms using the derived clusters based on their characteristics developed during the data enrichment process. The list of the specifications is as follows:

- i. Very close clusters of buildings of orthogonal shape, in which buildings are almost in the same orientation, depending on the orientation threshold, classified as VC-i ($i = 1, n$), enriched during the data enrichment process described in Section 5.4, are aggregated, squared, enlarged and simplified to form single buildings in the target scale with the aggregation algorithm with orthogonal sides. However, if buildings are not in exceptional positions such as corner hanging, corner touching or almost overhanging (see Figures 7.4 and 7.5, pages 203 and 204) in such clusters, the aggregation algorithm with non-orthogonal sides, which also preserves orthogonality of the original sides of the building outline, can also be applied depending on user choice.
- ii. Very close clusters of buildings of non-orthogonal shape, classified as VC-i ($i = 1, n$), are only aggregated with the algorithm developed for dealing with non-orthogonal shaped clusters.
- iii. Medium range clusters of buildings of orthogonal shape (buildings with almost similar orientation), classified as MS-i ($i = 1, n$), are also subject to the same specification of generalization as in (i) above.
- iv. Medium range clusters of buildings of non-orthogonal shape, classified as MS-i ($i = 1, n$), are subject to the same specification of generalization as in (ii) above.
- v. Medium range clusters comprising of buildings with large orientation differences, classified as ML-i ($i = 1, n$), are aggregated with the algorithm developed for dealing with non-orthogonal clusters.
- vi. Clusters of buildings with any of the above category labels, falling below the minimum threshold of dimension in terms of width and height based on the GOBR of each cluster, are symbolized by a square, preserving general orientation of the cluster.

- vii. All single buildings which get isolated from clusters during the clustering process are removed from the target scale if they have not been identified to be salient during the data mining process.

Tolerance values used during automatic generalization:

- i. Aggregation distance tolerance for the very close distance range clusters: 0.5mm of the target map scale.
- ii. Aggregation distance tolerance for the medium distance range clusters: the maximum distance threshold is 2mm of the target map scale.
- iii. Enlargement (exaggeration) of narrow sections and juts in the application of the aggregation algorithm with orthogonal sides: 0.5mm of the target map scale.
- iv. Simplification of building edges: 0.5mm of the target map scale.
- v. Inner holes less than an area of 1.5mm x 1.5mm of the target map scale are removed.
- vi. A cluster of buildings with an area less than 1mm x 1mm of the target map scale is symbolized with a squared symbol of dimensions 1mm x 1mm of the target map scale. Symbolization may create topological conflicts in the generalized data.

8.2.2 External validation of the results of generalization

Validation of the generalization results

Stoter *et al.* (2009) have established a robust framework for validating generalization results. This framework evaluates the generalized results using three methods: (a) qualitative evaluation by cartographic experts (b) automated constrained based evaluation and (c) visual comparison of different results. However, for the validation of the generalization results of the focus maps of the two test areas, a qualitative evaluation of the generalized results is carried out in comparison with the generalized results obtained from the proprietary ArcGIS software, which is the visual comparison of different results - method (c) - mentioned in this framework based on the preservation constraints

and the legibility constraints of the generalized objects as further described in Section 3.3.3, page 74.

Generalized results - I: Newham area

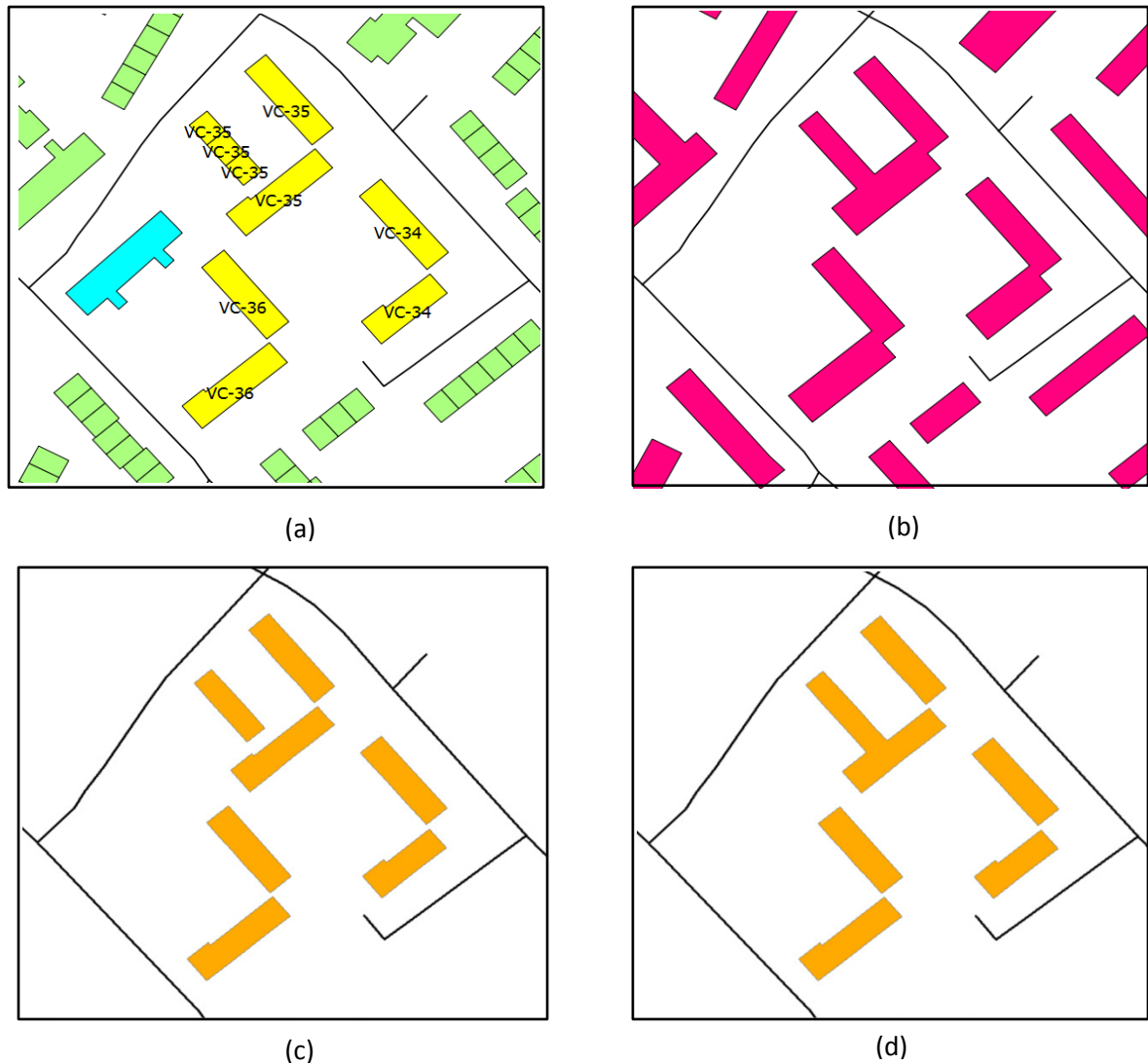


Figure 8.17 Generalization results of clusters of orthogonal shape: (a) three source clusters highlighted in yellow colour (b) generalized amalgams with a distance of 4m (c) generalized amalgams in the ArcGIS software with a distance of 4m and (d) generalized amalgams in the ArcGIS software with a distance of 16m.

Figure 8.17 (b) depicts the generalized amalgams of three clusters classified as ‘very close’ and orthogonal in shape using the aggregation algorithm specifically developed for treating the orthogonal clusters. Aggregation distance used by the algorithm is the distance used to classify ‘very close’ clusters (distance - 4m). None of the three clusters

are aggregated when using the polygon aggregation tool with the orthogonal shape preserving option in the ArcGIS software with the same distance as depicted in Figure 8.17(c). Even when the aggregation distance tolerance is increased to 16m which is four times the initial distance, only a couple of buildings in the cluster classified as ‘VC-35’ are aggregated (Figure 8.17(d)).

Generalized results - II: Newham area

Figures 8.18 and 8.19 depict the aggregated results of a cluster classified as ‘very close’ and ‘non-orthogonal in shape’, obtained from the generalization tool to deal with non-orthogonal shaped clusters in this research and the polygon generalization tool available in the ArcGIS software respectively.

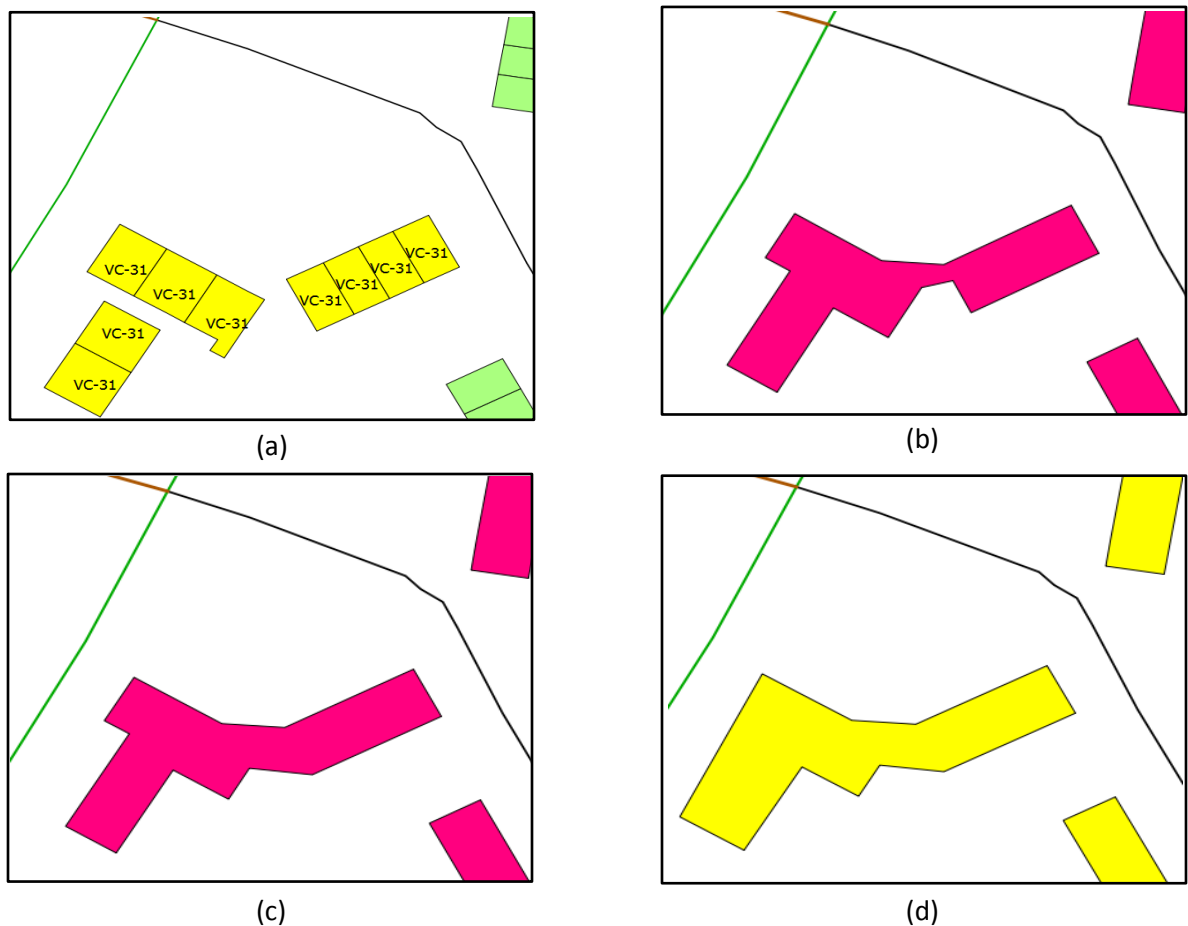


Figure 8.18 Generalization results of clusters of non-orthogonal shape with the research tool: (a) three source clusters highlighted in yellow colour (b) amalgam with an aggregation distance and a space triangle edge distance of 4m (c) amalgam with an aggregation distance of 4m and a space triangle edge distance of 8m and (d) amalgam in (c) after simplification with a tolerance of 4m.

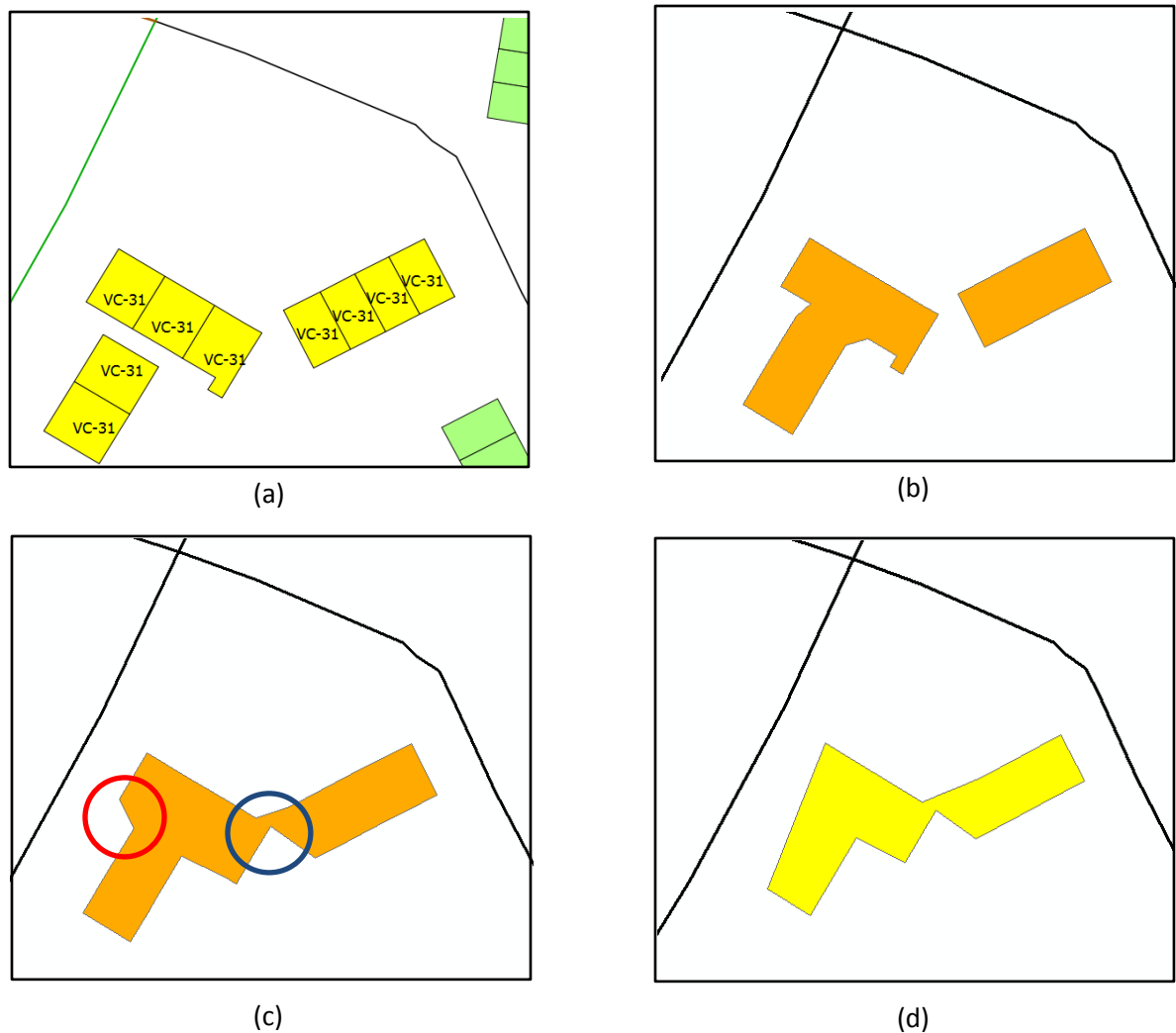


Figure 8.19 Generalization results of clusters of non-orthogonal shape with the ArcGIS software: (a) source clusters highlighted in yellow colour (b) amalgam with an aggregation distance of 4m (c) amalgam with an aggregation distance of 8m before simplification and (d) amalgam after simplification with a tolerance of 4m.

In Figure 8.18 (b) and (c), when observing the results of amalgams, it can be observed that the gaps between the buildings in the clusters have been exaggerated to improve legibility by increasing the edge distance threshold with the research tool. However, when observing the results in Figure 8.19(b), obtained from the ArcGIS software with an aggregation distance of 4m which is the clustering distance used to create ‘very close’ clusters in generating results, the cluster has been aggregated into two amalgams. However, when the distance is increased to double the cluster distance, the amalgam is created but the filled gap is still concave in shape, causing poor legibility (see the bridge

circled in blue colour in Figure 8.19(c)). Also, some squared edges have turned out to be non-orthogonal after aggregation (see the edges circled in red colour in Figure 8.19(c)).

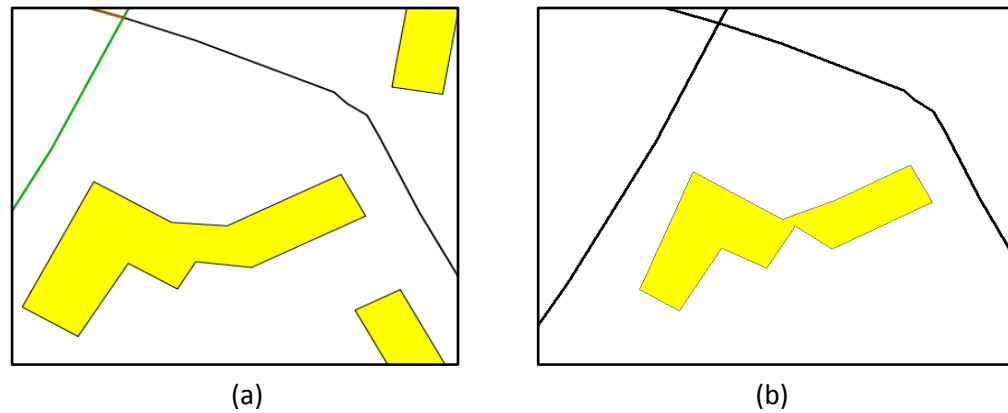


Figure 8.20 Comparison of amalgams of the non-orthogonal shaped cluster depicted in Figure 8.19(a) above: (a) created with the research tool and (b) created with the ArcGIS software.

Table 8.1 summarises the comparison of the generalized results given in Figure 8.17 (for orthogonal shaped clusters) and Figures 8.18, 8.19 and 8.20 (for non-orthogonal shaped cluster) in terms of preservation and legibility generalization constraints.

Table 8.1 Evaluation of the generalization constraints on amalgams of Newham data.

Results evaluated	Evaluation Criteria	Generalized method	
		Research tools	ArcGIS software
Generalized results - I in Figure 8.17 for orthogonal shaped clusters	Preservation constraints	General orientation, squareness and general shape well preserved	Partial amalgam creation based on the distance increment with a very small deviation of the shape from the initial data
	Legibility constraints	Dimensions of bridges between gaps exaggerated well- preserving uniformity with no conflicts	Gaps between buildings badly filled (only one gap bridged)
Generalized results - II in Figures 8.18, 8.19 and 8.20 for the non- orthogonal shaped cluster	Preservation constraints	General orientation and general shape well preserved	Shape badly preserved
	Legibility constraints	Gaps between buildings well bridged	Bad legibility in the gaps bridged

Generalized results - I: Tower Hamlets area

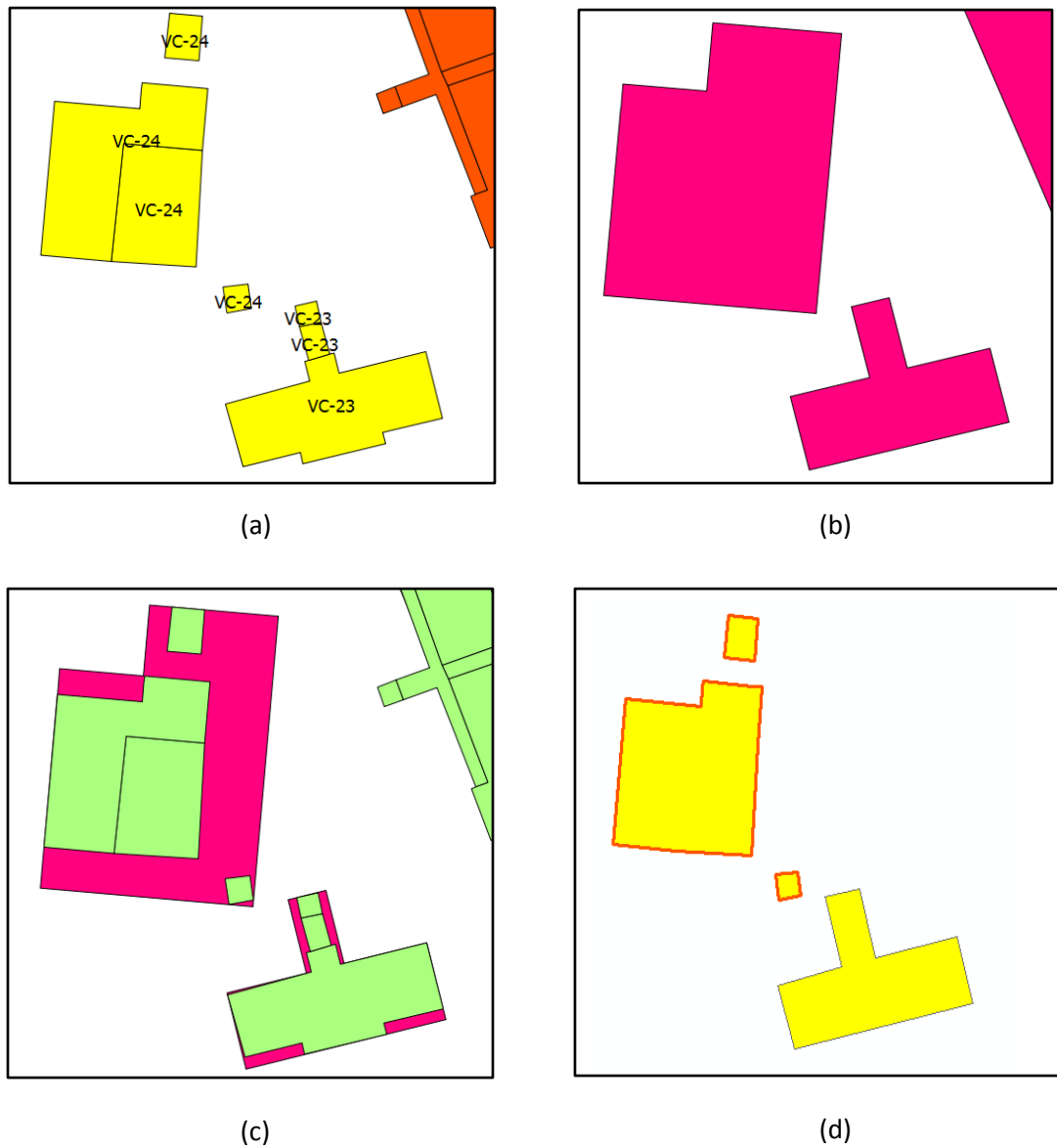


Figure 8.21 Generalization results of clusters of orthogonal shape: (a) two source clusters highlighted in yellow colour (b) amalgams with an aggregation, exaggeration and simplification distance of 4m with the research tool (c) amalgams in (b) overlaid with source clusters in (a), and (d) amalgams after simplification with a tolerance of 4m using the ArcGIS software.

Generalized results - II: Tower Hamlets area

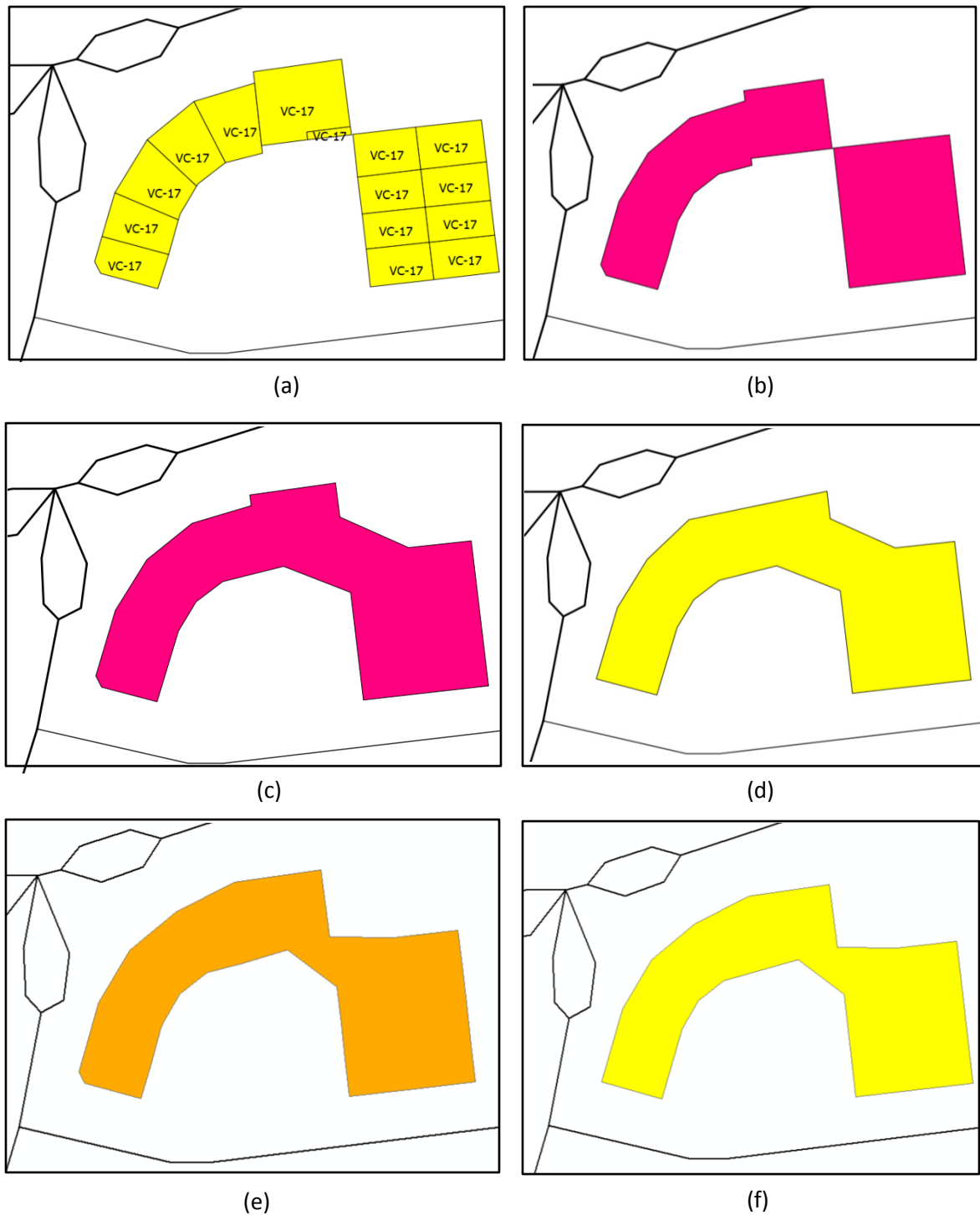
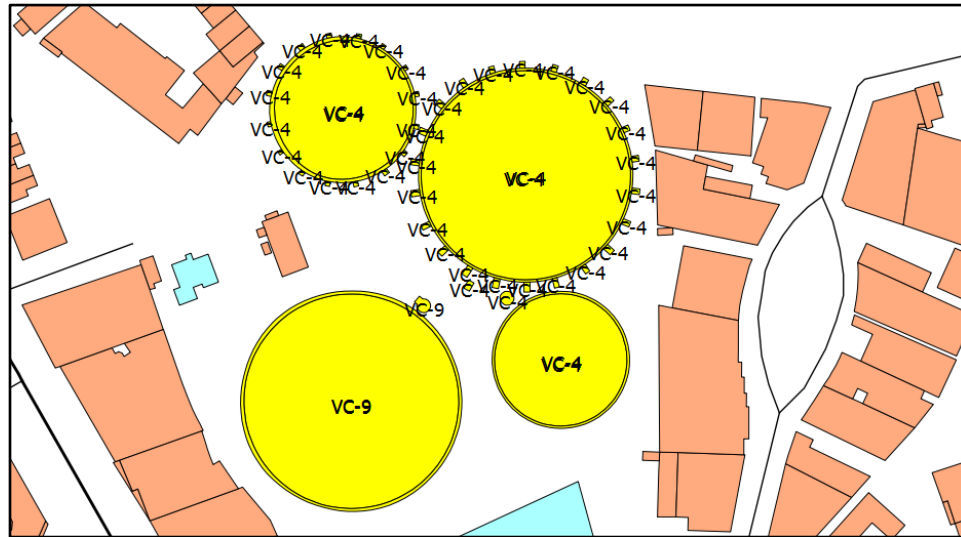


Figure 8.22 Generalization results of a cluster of non-orthogonal shape: (a) source cluster highlighted in yellow colour (b) amalgam with an aggregation distance of 4m and a space triangle edge threshold of 4m (c) amalgam before simplification with the two thresholds with a value of 8m (d) amalgam after simplification with 4m tolerance with the research tool (e) amalgam before simplification with a distance of 8m and (f) amalgam after simplification with a tolerance of 4m with the ArcGIS software.

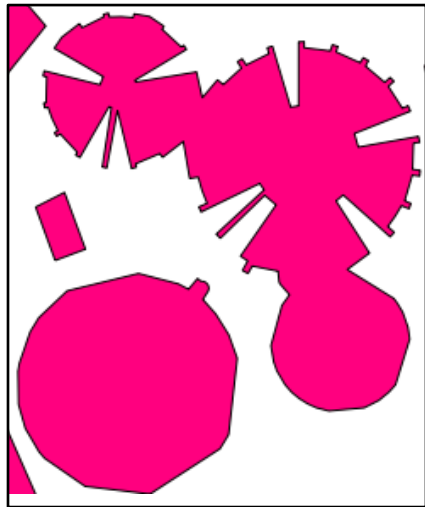
Figure 8.21(b) depicts generalized amalgams of the two clusters classified as ‘very close’ and orthogonal in shape using the aggregation algorithm specifically developed for treating the orthogonal clusters. Aggregation distance used by the algorithm is the distance used to classify ‘very close’ clusters (distance - 4m). When observing Figure 8.21 (c), the cluster labelled as ‘VC-23’ is aggregated, squared and exaggerated to create the amalgam, preserving general orientation. The long and narrow jut of this cluster has been exaggerated to meet the exaggeration threshold. However, when observing the results obtained from the ArcGIS software, the cluster with the label ‘VC-24’ is not aggregated with the polygon aggregation tool, together with the orthogonal shape preserving option. Only one cluster (cluster classified as ‘VC-23’) is aggregated with the ArcGIS tool as depicted in Figure 8.21 (d). The long and narrow jut of the cluster is only simplified since this tool does not have the facility to exaggerate narrow juts.

Figure 8.22 depicts the results of aggregation of the source cluster (Figure 8.22(a)) classified as ‘very close’ and ‘non-orthogonal in shape’ with the corner touching buildings. When it is aggregated with the aggregation tool to deal with non-orthogonal clusters developed in the research with an aggregation distance threshold and a space triangle edge distance threshold equal to the cluster distance (i.e. 4m), the amalgam is created solely by the buffer operation with a very narrow jut (see Figure 8.22 (b)). The reason is that the space triangles in the triangulation are not considered in this instance since the buffer operation with dilation and erosion brings out a polygon amalgam (not a multi-polygon which is a collection of polygons). In this situation, the aggregation distance and the space edge triangle distance are increased to double the clustering distance to create an improved amalgam (see Figure 8.22(c)). When aggregating with the ArcGIS tool with a 4m aggregation distance, no aggregation is created. When the distance is increased to 8m, an amalgam is created (see Figure 8.22(e)). However, when comparing the two simplified amalgams (final amalgams) created from both tools, the amalgam created by the research tool produces a uniform bridge (compare Figures 8.22 (d) and 8.22(f)).

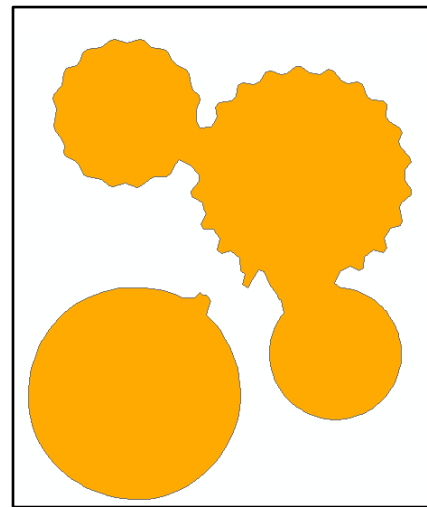
Generalized results - III: Tower Hamlets area



(a)



(b)



(c)

Figure 8.23 Generalization results of a cluster of non-orthogonal shape: (a) two source clusters highlighted in yellow colour (b) amalgams with an aggregation distance of 4m and a space triangle edge threshold of 4m with the research tool and (c) amalgams with an aggregation distance of 4m with the ArcGIS software, no simplification applied in both cases.

When applying the aggregation function for non-orthogonal shapes on the two circular shaped clusters depicted in Figure 8.23(a), the clusters are amalgamated, but with unusual shapes (Figure 8.23(b)). The reason is that in the dilation and erosion process, the edges of buffers are flattened with the buffer parameters used. Although this enables keeping

orthogonal edges of the buildings in the outline of the amalgam orthogonal, for circular shaped buildings, it tends to create irregular edges. When observing the results generated from the same clusters using the ArcGIS software (Figure 8.23(c)), its building aggregation algorithm too does not support merging circular shaped clusters.

Table 8.2 Evaluation of the generalization constraints on amalgams of Tower Hamlets data.

Results evaluated	Evaluation Criteria	Generalized method	
		Research tools	ArcGIS software
Generalized results - I in Figure 8.21 for orthogonal shaped clusters	Preservation constraints	General orientation, squareness and general shape well preserved	No amalgam created for the cluster with both the attached and the detached buildings Only amalgam created for the cluster with attached buildings, well-preserving orientation and shape
	Legibility constraints	Dimensions of bridges between gaps exaggerated well- preserving uniformity with no conflicts	Poor legibility due to narrow jut of the amalgam of the attached building cluster
Generalized results - II in Figure 8.22 for the non-orthogonal shaped cluster	Preservation constraints	General orientation and general shape well preserved	General orientation and general shape well preserved
	Legibility constraints	Gap between buildings well bridged	Poor legibility in the bridged gap
Generalized results - III in Figure 8.23 for the non-orthogonal shaped cluster	Preservation constraints	Shape preservation is bad	Shape preservation is bad
	Legibility constraints	Gaps between buildings fairly bridged	Gaps between buildings fairly bridged

8.2.3 External validation of the results of landmark saliency

Deriving landmark saliency with the J48 implementation

During the salient landmark derivation process with the J48 implementation, the sensitivity analysis and the attribute discretization (transformation) were carried out in each region as performed on the test data sets as described in Section 6.2.2, pages 182 and 185. None of the salient landmarks were identified at the topmost level (i.e. the zero level in the tree) in each of the regions in both data sets. However, all the salient landmarks were identified at the next level (level 1 on the tree) in each of the regions in both test areas.

It is important to mention that the landmark saliency results obtained from the J48 implementation within a region is not similar to the results obtained by applying it on the sub-regions of a particular region even if the same attribute discretization and the sensitivity analysis are carried out on the data (Figure 8.24).

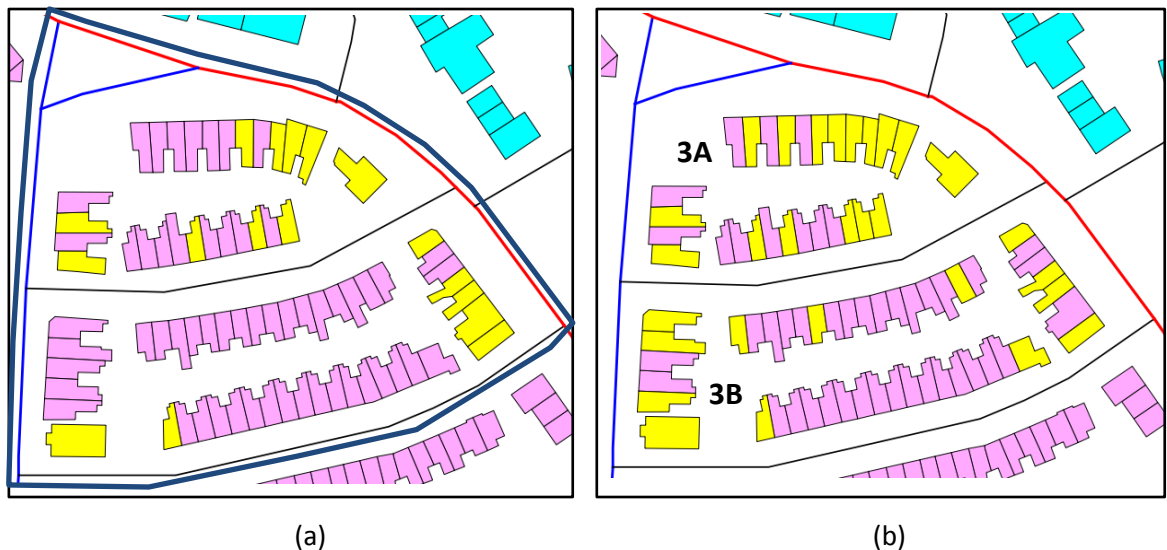


Figure 8.24 (a) Salient landmarks (highlighted in yellow colour) derived from region 3 of Newham area (see Figure 8.9), delineated in blue and (b) salient landmarks (highlighted in yellow colour) derived in sub-regions 3A and 3B of the main region 3, applying the J48 implementation in each sub-region separately.

The J48 implementation has derived 17 salient buildings in region 3, 16 salient buildings in region 3A and 13 salient building in region 3B. The total landmarks derived using sub-regions within region 3 is 29. This is more than the salient buildings derived when applying the J48 implementation in the whole of region 3. Among the 17 salient buildings in region 3, 16 buildings are included in the 29 salient buildings derived from the sub-regions. These results emphasise that when applying the J48 implementation in a region, the sensitivity analysis and the attribute discretization must be carried out by taking into account the characteristics of the particular region only. Thus, generating results with the J48 implementation is region dependent.

Validation with the framework by Nothegger, Winter and Raubal (2004)

A quantitative evaluation of the results based on the framework of Nothegger, Winter and Raubal (2004), which is the extended model by Raubal and Winter (2002), implemented in this research, are carried out in regions to validate landmark saliency of the focus maps in the two test areas obtained with the J48 implementation of the C4.5 decision tree algorithm evaluated in Section 6.2.4, page 195. This framework derives the most salient landmark with the use of building facades with their visual and semantic characteristics at decision points. In this method, the building with the highest overall significance score is chosen as the most salient landmark. An individual significance score for each building at a decision point is calculated based on the significance value of each attribute falling in both semantic and visual characteristics of buildings for this purpose (see Table 8.3). A value of significance for each attribute under each of the two characteristics is calculated by the following formulae (4) and (5), based on the median absolute deviation (MAD) which is a statistical measure.

$$\text{Score} = (|x - \text{med}(x)|) / \text{MAD}(x) \quad (4)$$

$$\text{MAD}(x) = \text{med}(|x - \text{med}(x)|) / 0.6745 \quad (5)$$

In these equations, x is a value of an attribute, $\text{med}(x)$ is the median of all values of an attribute of all the buildings considered at the decision point and $\text{MAD}(x)$ is the MAD from the median.

Table 8.3: Method of calculating the overall significance score of each building using individual significance value of each attribute based on the framework by Nothegger, Winter and Raubal (2004). α_1 , α_2 and α_3 are the scores of the three attributes in the visual category, β_1 and β_2 are the scores of the two attributes in semantic category, and w_1 and w_2 are weights assigned to visual and semantic categories respectively.

bid	Significance score of each attribute in Visual category			Significance score of each attribute in Semantic category		Significance sub score: Visual	Significance sub score: Semantic	Overall significance
	α_1	α_2	α_3	β_1	β_2	$\sigma_1 = (\alpha_1 + \alpha_2 + \alpha_3) / 3$	$\sigma_2 = (\beta_1 + \beta_2) / 2$	$(\sigma_1 * w_1 + \sigma_2 * w_2) / (w_1 + w_2)$
1	α_1	α_2	α_3	β_1	β_2	$\sigma_1 = (\alpha_1 + \alpha_2 + \alpha_3) / 3$	$\sigma_2 = (\beta_1 + \beta_2) / 2$	$(\sigma_1 * w_1 + \sigma_2 * w_2) / (w_1 + w_2)$
2	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-

Although all types of attribute values such as numeric and text can be handled by the J48 implementation, in order to apply text values in the framework of Nothegger, Winter and Raubal (2004), the text values of the attribute ‘orientation to road’ need to be transformed into ordinal values given below. The values of the other attributes given in Table 6.1, page 166 are not required to be transformed when used in this framework.

- Orientation to road: {Corner, Angular, Parallel, Across, None} => {1, 2, 3, 4, 5}

In addition, the attributes removed based on the sensitivity analysis in processing the results with the J48 implementation are ignored in generating results with the framework in order to keep the consistency. Further, in the use of the framework, equal weights are assigned to the three visual, structural and semantic characteristics of the attribute values.

When this model was applied to the test data in region 5 of Newham area (Figure 8.9(b)), out of seventy one (71) buildings, fifty (50) buildings were assigned an overall score of infinity. Further testing the values of data, it was observed that the MAD values of some attributes of these 50 buildings became zero in the calculation, leading the overall score to become infinity because the MAD value was in the denominator in the equation (4) above used to measure the saliency score. Further, the score used by Nothegger, Winter and Raubal (2004) was the Z-score, a multiplicative factor of the standard deviation used to identify values around the mean of a normally distributed data set. Therefore, in cases

where their score became infinity, it was considered to be zero in applying their model in the validation process. When investigating the total sub-score of each of the visual, structural and semantic characteristics (see Table 8.3), which was the arithmetic mean of all the scores of objects under each characteristic, the sub-scores generalized the important values of these characteristics, leading to a doubtful situation over which attributes and/or their values, a certain object became salient. Also, the overall score which was the weighted mean of the three sub-scores, tended to assign higher values. Nothegger, Winter and Raubal (2004) have used data transformation before applying this score on the data to eliminate this effect to make the distribution symmetric. However, this approach would in turn lose important outliers in the original data.

The reason why Nothegger, Winter and Raubal (2004) have developed this score by extending the model of Raubal and Winter (2002), which uses the standard deviation around the mean to derive saliency score, is that it can deal with outliers that have a strong impact when using the standard deviation around the mean especially when data are not normally distributed as further emphasised by Leys *et al.* (2013). However, they have used this equation to derive landmark saliency only on a few decision points, considering the visual and the semantic characteristics of building facades, excluding the structural characteristics.

According to Leys *et al.* (2013), the MAD is a robust measure to detect outliers especially from the non-normally distributed data because of its non-sensitivity to outliers, although Nothegger, Winter and Raubal (2004) have used a scalar ($1/0.6745$) with the MAD so that the score approximates a Z - score of normally distributed data. Considering all these aspects, the validation of landmark saliency for the test data in this research is done with two methods: (a) using the framework by Raubal and Winter (2002) to investigate how the outliers are impacted by using the standard deviation around the mean of the test data and (b) a new method is developed to detect outliers to derive salient features with the use of the MAD on the decision criterion proposed by Leys *et al.* (2013). This method is discussed in detail on page 285.

Validation with the framework by Raubal and Winter (2002)

Similarly as for validation of the framework of Nothegger, Winter and Raubal (2004), text values of the attribute ‘orientation to road’ are required to transform into ordinal values. The values of other attributes given in Table 6.1, page 166 are not required to be transformed when used in this framework. In addition, the attributes removed based on sensitivity analysis in processing the results with the J48 implementation are ignored in generating results with this framework in order to keep the consistency. Further, in the use of this framework, equal weights are assigned to the three visual, structural and semantic characteristics of the attribute values.

In the framework of Raubal and Winter (2002), the attributes of visual, semantic and structural characteristics are tested for their difference from the local mean using the standard deviation in detecting outliers. In this method each building is subject to a hypothesis test that determines whether an attribute value of each of the three characteristics (measures) of a building is significant with a binary value of 1, or is insignificant with a binary value of 0 using the significance of deviations from the local mean. Table 8.4 describes how the total significance score is calculated on a building using the framework of Raubal and Winter (2002).

Table 8.4 Deriving a total significance score for a particular building by the method of Raubal and Winter (2002).

Measure	Attribute	Attribute value	Significance	Significance score	Weight	Weighted significance	Total significance
Visual	α_1	-	1	$S_{vis} = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) / 4 = 0.75$	$W_{vis} = 1$	$S_{vis} \times W_{vis} = 0.75$	1.75
	α_2	-	0				
	α_3	-	1				
	α_4	-	1				
Structural	β_1	-	1	$S_{str} = (\beta_1 + \beta_2)/2 = 0.5$	$W_{str} = 1$	$S_{str} \times W_{str} = 0.5$	
	β_2	-	0				
Semantic	Υ_1	-	0	$S_{sem} = (\Upsilon_1 + \Upsilon_2) / 2 = 0.5$	$W_{sem} = 1$	$S_{sem} \times W_{sem} = 0.5$	
	Υ_2	-	1				

Tables 8.5 and 8.6 depict the evaluation results of salient landmarks based on the framework of Raubal and Winter (2002). A statistical significance measure is derived in this approach using a decision criterion which is plus or minus 2.5 (Z – score equivalent to 98% confidence interval) of the standard deviation around the local mean, assuming that the data are normally distributed, for all the attributes of each building in a specific region depicted in Figures 8.9 and 8.13 on pages 257 and 261 respectively. This particular value of the Z-score is used due to the fact that using an interval with a mean plus or minus a Z – score of either 2, 2.5 or 3 around the standard deviation is a common practice in detecting outliers (Leys *et al.*, 2013; Miller, 1991), although the choice of this value in detecting outliers is subjective. This research uses the middle value of 2.5 to detect values of outliers that are outside both limits of the interval (to detect outliers with the low and the high values) in the decision criterion.

Table 8.5 Evaluation of landmark saliency of the focus map with the framework of Raubal and Winter (2002) in the regions depicted in Figure 8.9, page 257 of Newham area.

Region	No. of buildings	Landmarks identified by J48	Landmarks identified by the framework with total significance > 0	Landmarks matched with the framework	Landmarks mismatched with the framework	Percentage of landmarks matched with the framework
1	311	42	57	27	15	64%
2	214	32	39	26	6	81%
3	73	17	20	13	4	76%
4	322	19	34	18	1	95%
5	72	28	15	14	14	50%
6	406	38	43	13	25	34%
7	285	16	20	9	7	56%

Table 8.6 Evaluation of landmark saliency of the focus map with the framework of Raubal and Winter (2002) in the regions depicted in Figure 8.13, page 261 of Tower Hamlets area.

Region	No. of buildings	Landmarks identified by J48	Landmarks identified by the framework with total significance > 0	Landmarks matched with the framework	Landmarks mismatched with the framework	Percentage of landmarks matched with the framework
1	149	33	35	16	17	48%
2	206	32	45	14	18	44%
3	123	36	30	25	11	69%
4	237	64	52	39	25	61%
5	93	41	30	22	19	54%

When analysing the results obtained from this framework, a building with a total significance value greater than zero is identified to be a feature with some salience for a landmark. It is also important to note that the selection of a threshold value to detect a salient landmark is a subjective process where a value is to be determined for the whole area or for a specific region by investigating the results.

When observing the results of the derived salient landmarks in the Newham area (Table 8.5), 50% or more of the results match with that of obtained using the framework of Raubal and Winter (2002). However, matching percentages vary from one region to the other (see the visual comparison of the results of region 5 in Figure 8.25). The main reason that the results of the J48 implementation are not matched with that of the framework consistently is due to the two methods working in the opposite direction in detecting outliers. In the J48 implementation, data are subjected to a process of transformation with an equal binning approach which is very sensitive in detecting outliers as mentioned in Section 6.2.2, page 185. The reason is that it makes data asymmetric to detect outliers. When the decision criterion for outliers is chosen in the framework, it is assumed that the distribution of the data is normal. It is likely that the mean and the standard deviation of each attribute is strongly impacted by outliers thus affecting the interval of the decision criterion, causing building objects not to be selected as outliers. This is also impacted by the sample size of the data set and their varying values in each region as emphasised by the poor results indicated in the region 6. This is further emphasised by the comparison of the J48 results in the main region with its sub-regions discussed under the subsection - deriving landmark saliency with the J48 implementation - on page 277.

When observing the results of the derived salient landmarks in Tower Hamlets area (Table 8.6), the results are not as good as for Newham (only 45% or more are matched with the results of the J48 implementation) due to the same reasons mentioned above. Visual comparison of the results of region 1 of Tower Hamlet area is depicted in Figure 8.26.



Figure 8.25 Landmark saliency results (highlighted in yellow colour) of region 5 of Newham area: (a) results by the J48 implementation and (b) results by the framework of Raubal and Winter (2002).

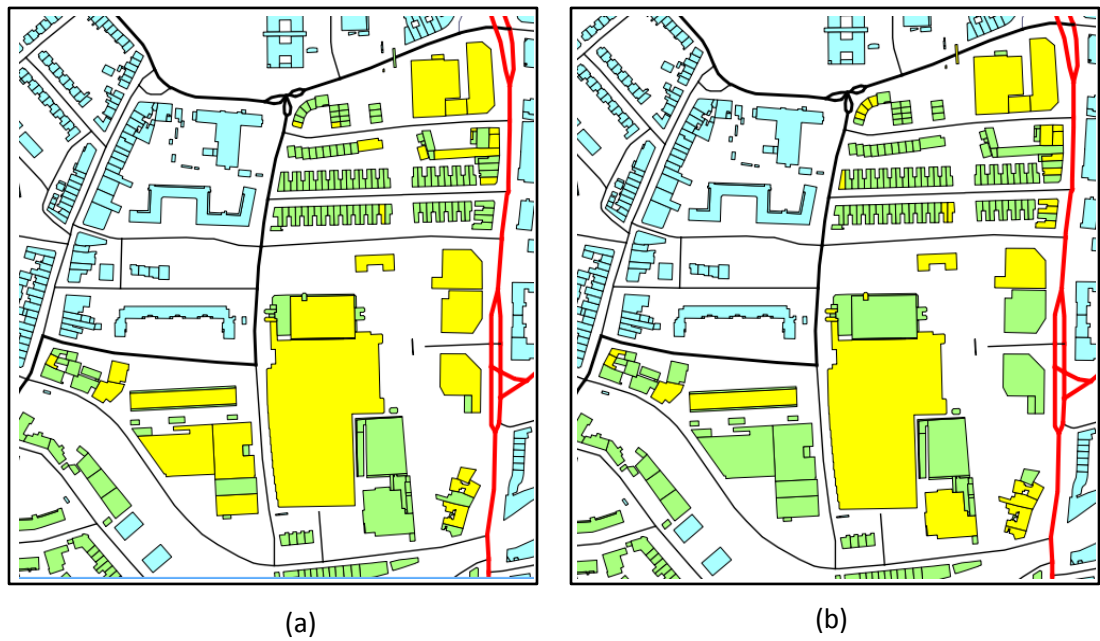


Figure 8.26 Landmark saliency results (highlighted in yellow colour) of region 1 of Tower Hamlets area: (a) results by the J48 implementation and (b) results by the framework of Raubal and Winter (2002).

Validation of the results with the method developed in this research

The new measure of salience is developed in this research using the MAD based on the decision criterion proposed by Leys *et al.* (2013) (see Appendix F.6 for the prototype). According to Leys *et al.* (2013), the MAD is defined as follows, citing Huber (1981).

$$\text{MAD} = b \text{ median}_i (|X_i - \text{median}_j (X_j)|) \quad (6)$$

Where, b is a constant (scalar), X_i is an attribute value of an object, X_j is the n original observations and median_i is the median of the series. According to Leys *et al.* (2013), b is usually a constant of 1.4826 used to assume normality of data, which is the scalar inverse of 0.6745 used by Nothegger, Winter and Raubal (2004) to make data asymptotically equal to the standard deviation of the standard normal distribution.

The criterion used to detect outlier values in this research is based on the following equation:

$$X_i > \text{median} + 2.5 * \text{MAD} \quad (7)$$

Where X_i is an attribute value of an object and b is assigned the same value of 1.4826 in the equation (6) for the assumption of normality of the data.

In detecting outliers according to this equation (7), only higher values of a particular attribute of an object are chosen as outliers. If a value of a particular attribute of a building is an outlier, such an attribute is assigned the binary value 1 and the value 0 is assigned if it is not an outlier in calculating the overall score of all the attributes of an object.

The reason to detect landmark saliency based on the values that are above the upper region of the decision criterion is that the values below the lower limit tend to stand out as building features that have no discerning values to become salient (e.g. buildings of smaller size) as observed from the results obtained with this method. Further, the higher the values of attributes such as the minimum distance to road, neighbourhood density and elongation used in this research, the lower is the ability to identify a building as salient with these attributes (for example, according to elongation defined in Table 6.1, page 166, the higher the value of elongation index, the more square is the building). Therefore, before applying the criterion based on the upper limit to calculate a salience score, all values of these three attributes are transformed by getting the inverse of the values so that the lower values of these attributes tend to stand out (see attributes 5, 6 and 7 in Table 8.7). Then, the ordinal values used in this research for attributes - 'diversely oriented edges', 'orthogonality index' and 'building importance' - as described in Section 6.1.1, pages 168 and 179 and attribute - 'orientation to road' - (Section 6.11, page 170) with nominal values (see Figure 6.5, page 177) are transformed to true or false {0,1} binary ordinal values and not included in the statistical significance measure since they do not tend to be sensitive (as outliers) in the decision criterion used with the MAD (see attributes 1, 2, 3 and 4 in Table 8.7). Instead, such binary ordinary values are combined with the salience measure given by the numeric values in terms of a binary value as explained in Table 8.8. Apart from the attributes given in Table 8.7, all the other attributes given in Table 6.1, page 166, are used in this new method with no transformation.

Table 8.8 shows the method of deriving an individual score for each characteristic (measure), as well as the total weighted significance. The significance of attributes that is highlighted in the table (also see Table 8.7 for their details) is directly derived and thus not subjected to the statistical test with the MAD because they were binary {0, 1} values as depicted in Table 8.7. All the other values are assigned either a value of 1 or 0 based on the statistical test using the formula (7).

Table 8.7 Attributes and their transformed values to be compatible with the new landmark saliency measure.

#	Attribute	Description	Transformed value(s)
1	Building importance (Priority) (Y1)	Attraction (Cultural and Historical, Botanical and Zoological, Recreational, Tourism, Pubs and Retail shops)	1
		Health	
		Educational	
		Public Infrastructure	
		Transport	
		Sports and Entertainment	
		Commercial	0
		Manufacturing	
		Residential	
2	Orientation to road (β_4)	Corner or angular	1
		Parallel or across	0
3	Diversely oriented edges (α_6)	Yes	1
		No	0
4	Orthogonality index(α_5)	Non orthogonal	1
		Orthogonal	0
5	Minimum distance to road	Distance between building and the closest road (DST)	1 / DST
6	Elongation	Ratio between width / length (e)	1 / e
7	Neighbourhood density	Ratio between number of buildings divided by the area of the region around a building (d)	1 / D

Table 8.8 Deriving a total significance measure for landmark saliency for a building to be applied to the new method based on the MAD.

Measure	Attribute	Value	Significance of attribute	Total significance of each measure (Individual significance measure)	Weight	Total Weighted significance
Visual attraction	α_1	---	S_{α_1}	$S_{vis} = S_{\alpha_1} + S_{\alpha_2} + S_{\alpha_3} + S_{\alpha_4} + S_{\alpha_5} + S_{\alpha_6}$	W_{vis}	$\frac{[(S_{vis} * W_{vis}) + (S_{str} * W_{str}) + (S_{sem} * W_{sem})]}{[W_{vis} + W_{str} + W_{sem}]}$
	α_2	---	S_{α_2}			
	α_3	---	S_{α_3}			
	α_4	---	S_{α_4}			
	α_5	---	S_{α_5}			
	α_6	---	S_{α_6}			
Structural attraction	β_1	---	S_{β_1}	$S_{str} = S_{\beta_1} + S_{\beta_2} + S_{\beta_3} + S_{\beta_4} + S_{\beta_5} + S_{\beta_6} + S_{\beta_7}$	W_{str}	
	β_2	---	S_{β_2}			
	β_3	---	S_{β_3}			
	β_4	---	S_{β_4}			
	β_5	---	S_{β_5}			
	β_6	---	S_{β_6}			
	β_7	---	S_{β_7}			
Semantic attraction	γ_1	---	S_{γ_1}	$S_{sem} = S_{\gamma_1}$	W_{sem}	

α_1 – DEM height, α_2 – size, α_3 – Number of corners, α_4 – Inverse of elongation, α_5 – orthogonality index and α_6 – diversely oriented edges, β_1 – Inverse of minimum distance to road, β_2 – No. of adjacent neighbours, β_3 – orientation to North, β_4 – Orientation to road, β_5 – Average orientation to neighbours, β_6 – Minimum distance to neighbour and β_7 – Inverse of neighbourhood density and γ_1 – Importance (priority).

In this method, by observing the individual significance score of each measure, it can be directly identified how many attributes have contributed to making a particular building salient. The combination of the total weighted significance and the individual significance measure is used to decide the criteria to be formulated to make a particular building a salient landmark. However, the total significance measure can be calculated with different weights depending on the context of wayfinding (mode of travel or according to the user preference).

The criteria used in this research to derive a salient landmark with this new method are as follows:

A building becomes salient if its total weighted significance > 1 and at least the sum of individual scores of the two measures (characteristics) ≥ 2 (equal weight of 1 is assigned in this research). In applying the criteria, the attributes removed based on the sensitivity analysis on the J48 implementation are ignored in generating results using the framework in order to keep the consistency. Further, in the use of the framework, equal weights are assigned to the three visual, structural and semantic characteristics of the attribute values.

Tables 8.9 and 8.10 give a summary of the results of the J48 implementation in the two test data sets matched with the new method developed based on the MAD.

Table 8.9 Evaluation of landmark saliency with the new method developed on the MAD in the Newham data set.

Region	No. of buildings	Landmarks identified by J48	Landmarks identified by the new method	Landmarks matched with the new method	Landmarks mismatched with the new method	Percentage of landmarks matched with the new method
1	311	42	16	11	31	26%
2	214	32	28	20	12	63%
3	73	17	4	2	15	12%
4	322	19	15	9	10	47%
5	72	28	11	11	17	39%
6	406	38	39	26	12	68%
7	285	16	9	7	9	44%

Table 8.10 Evaluation of landmark saliency with the new method developed on the MAD in Tower Hamlets data set.

Region	No. of buildings	Landmarks identified by J48	Landmarks identified by the new method	Landmarks matched with the new method	Landmarks mismatched with the new method	Percentage of landmarks matched with the new method
1	149	33	16	9	24	28%
2	206	32	39	8	24	25%
3	123	36	16	13	23	36%
4	237	64	44	39	25	61%
5	93	41	14	13	28	32%

Figures 8.27 and 8.28 give a visual comparison of the results in region 5 of Newham area and region 1 of Tower Hamlets area respectively. When analysing the results, it is found that the percentage of the matching results between the J48 implementation and the new method has further dropped down in comparison to the matching between the same J48 implementation and the framework by Raubal and Winter (2002). This is mainly due to the following couple of reasons:

- Attribute values used in the J48 implementation to generate results are transformed before applying the new method as given in Table 8.7.
- A constraint is applied (see page 289) in the new method to make a landmark salient with the sum of the individual significance of the two measures (properties). This is to particularly to avoid a building with ‘priority’ value 1 (the only semantic measure) being qualified as a salient feature even if there is no significance based on the two other visual and structural properties.



Figure 8.27 Landmark saliency results (highlighted in yellow colour) of region 5 of Newham area: (a) results by the J48 implementation and (b) results by the new method on the MAD developed in this research.



Figure 8.28 Landmark saliency results (highlighted in yellow colour) of region 1 of Tower Hamlets area: (a) results by the J48 implementation and (b) results by the new method on the MAD developed in this research.

The validation of landmark saliency is further evaluated using the graphical visualisation of specific decision points representing both test areas with the Google street view. A crosscheck of the evaluation is carried out with the focus map results obtained from the J48 implementation, the framework by Raubal and Winter (2002) and the new method developed on the MAD in this research.

In Figures 8.30 to 8.33, each example is first shown in an inset on the focus map in Figure 8.29 (Inset A depicts the map in Figure 8.30, B depicts the map in Figure 8.31, C depicts the map in Figure 8.32 and D depicts the map in Figure 8.33), giving generalized background and salient landmarks at specific locations in Newham area, along with the photographs from the Google street view depicting the numbered salient landmarks. The calculations of the salient landmarks using the framework of Raubal and Winter (2002) and the new method developed using the MAD in this research are also given.

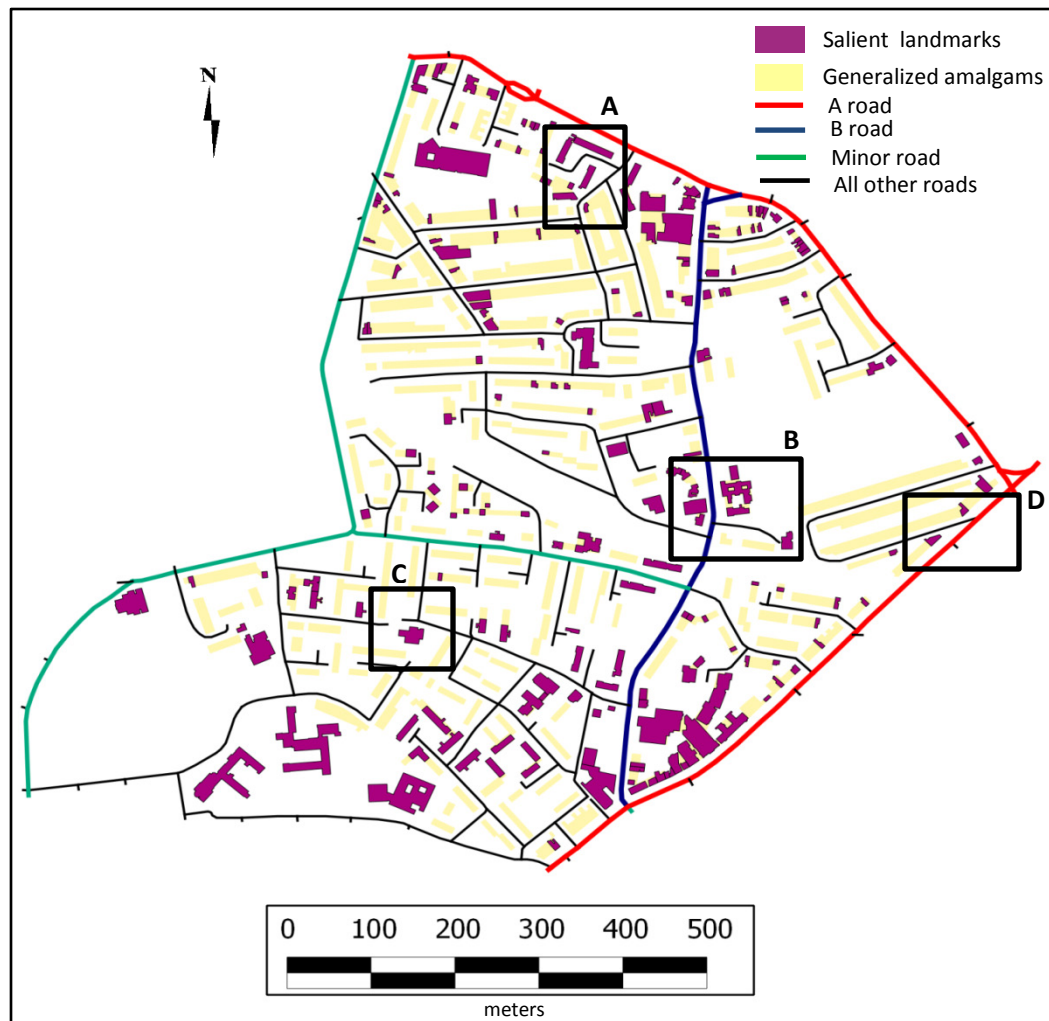
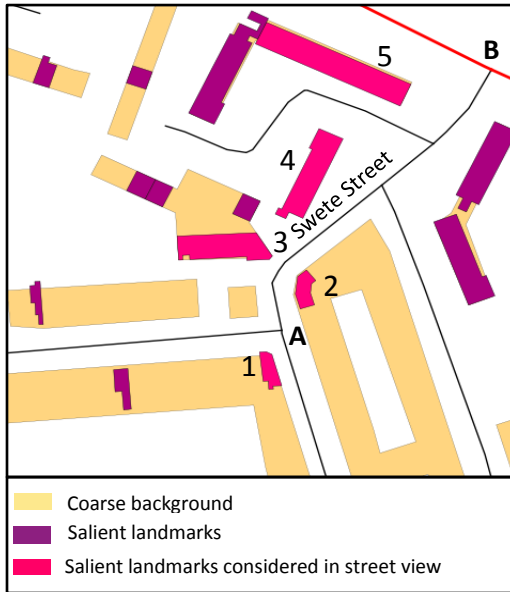


Figure 8.29 Focus map of Newham area with the insets A, B, C and D of the specific locations considered in the validation of the landmark saliency. Note: Map is not printed to scale.

Salient landmark results - I: Newham area



(a)



(b)



(c)



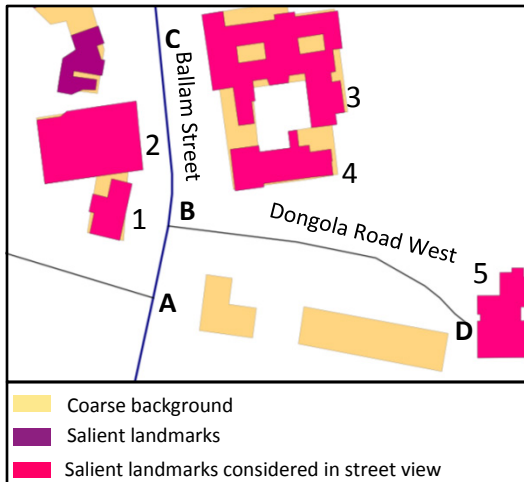
(d)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		$\alpha1$	$\alpha2$	$\alpha3$	ϵ	$\alpha1$	$\alpha2$	$\alpha3$	ϵ
1	Orientation: 353 ⁰	0.40	0	0	0.40	2	1	0	1
2	Orientation to road: Parallel	0.33	0	0	0.33	3	2	0	1.67
3	DEM height: 7m	0.40	0.20	0	0.60	3	4	0	2.33
4	DEM height: 16m	0	0.20	0	0.20	2	3	0	1.67
5	DEM height: 20m	0.20	0	0	0.20	2	1	0	1

(e)

Figure 8.30 Salient landmark visualization on Swete Street: (a) map (b) view towards A to B (c) view towards B to A (d) view towards A to B on the inset map and (e) calculations of salient landmarks. $\alpha1$: visual significance, $\alpha2$: structural significance, $\alpha3$: semantic significance and ϵ : total significance.

Salient landmark results - II: Newham area



(a)



(b)



(c)



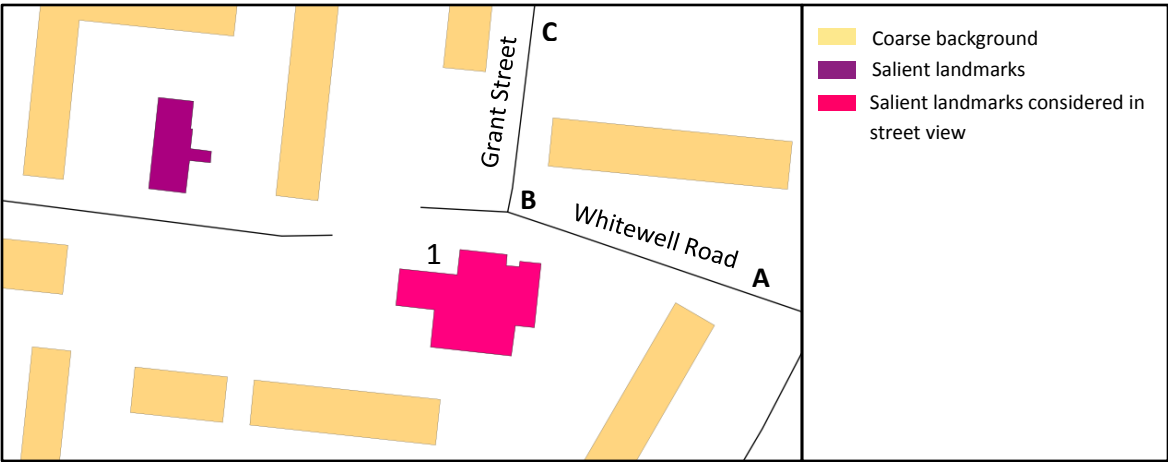
(d)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		α_1	α_2	α_3	ϵ	α_1	α_2	α_3	ϵ
1	Neighbourhood density: 0.0013m^{-2}	0	0.14	0	0.14	2	4	1	2.33
2	Neighbourhood density: 0.0014m^{-2}	0.17	0	1	1.17	1	3	1	1.67
3	Size: 645m^2	0.4	0.25	0	0.65	2	2	0	1.33
4	DEM height: 10m	0.4	0.25	0	0.65	2	3	0	1.67
5	DEM height: 43m	0.6	0.25	0	0.85	3	3	0	2

(e)

Figure 8.31 Salient landmark visualization on Ballam Street and Dongola Road West: (a) map (b) view towards A to B (c) view towards B to C (d) view towards B to D on the inset map and (e) calculations of salient landmarks. α_1 : visual significance, α_2 : structural significance, α_3 : semantic significance and ϵ : total significance.

Salient landmark results - III: Newham area



(a)



(b)



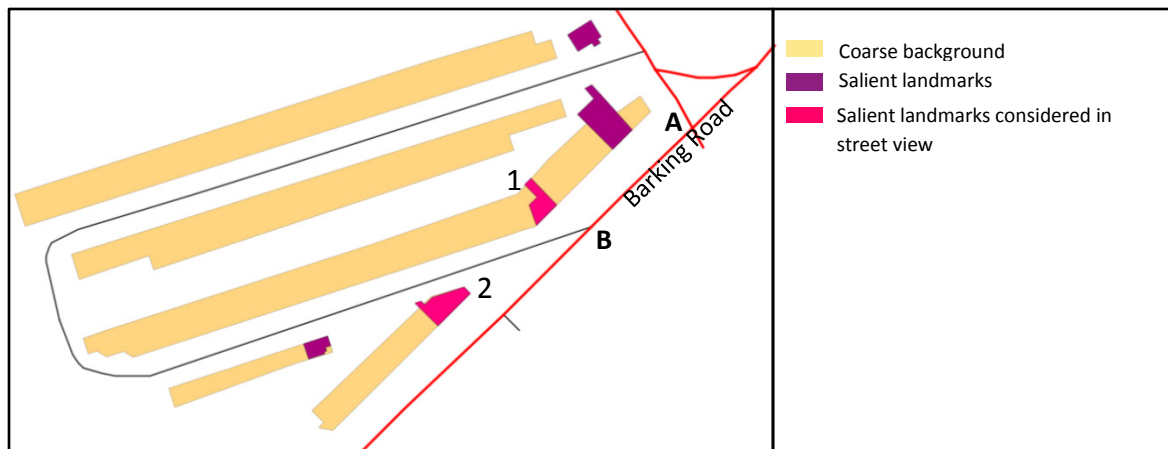
(c)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		$\alpha1$	$\alpha2$	$\alpha3$	ϵ	$\alpha1$	$\alpha2$	$\alpha3$	ϵ
1	Size: 442m ²	0	0.2	1	1.2	2	3	1	2

(d)

Figure 8.32 Salient landmark visualization on Whitewell Road and Grant Street: (a) map (b) view from A to B (c) view from C to B on the inset map and (d) calculations of salient landmarks. $\alpha1$: visual significance, $\alpha2$: structural significance, $\alpha3$: semantic significance and ϵ : total significance.

Salient landmark results - IV: Newham area



(a)



(b)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		α_1	α_2	α_3	ϵ	α_1	α_2	α_3	ϵ
1	Orientation to road: 'angular'	0	0.25	1	1.25	1	2	1	1.33
2	Diverse sides: 'yes'	0.40	0	1	1.40	4	2	1	2.33

(c)

Figure 8.33 Salient landmark visualization on Barking Road: (a) map (b) view from A to B on the inset map and (c) calculations of salient landmarks. α_1 : visual significance, α_2 : structural significance, α_3 : semantic significance and ϵ : total significance.

Salient landmark results - I: Newham area

All five (5) buildings on the Swete Street in Figure 8.30(a) identified as salient landmarks by the J48 implementation have significance scores with the framework of Raubal and Winter (2002) used for validation, implying such buildings to be landmarks. However, structural characteristics of discerning attributes of buildings with IDNs 1 and 2 emphasised by the J48 implementation do not provide any direct clue to be prominent when viewing the Google street view.

The characteristics used to identify a particular building as a landmark with these three methods are not the same (For example, J48 uses structural characteristic - orientation - while the framework uses visual characteristics in the case of buildings with IDNs 1 and 2). The new method based on the MAD identified both visual and structural characteristics of all landmarks as significant. The framework has not identified any visual characteristic of the building with IDN 4 while the method based on the MAD has given a visual significance for this building. This could be further appreciated when viewing the street view (Figure 8.30(d)) that the building with IDN 4 stands out because of its height. The reason is that the framework does not identify its height value as an outlier because of its less sensitivity to outlier detection.

According to the landmark saliency on the Google street view images of these buildings along the road, when moving from A to B as shown in Figure 8.30(a), buildings with IDNs 3 and 4 in the Figure 8.30(a) appear to be landmarks. When travelling in the opposite direction from B to A, building with IDN 1 appears to be a landmark in Figure 8.30(c). All these three buildings have been identified as salient landmarks by the method based on the MAD. The method on the MAD has not identified buildings with IDNs 1 and 5 as landmarks because of their fewer structural characteristics.

However, all the three methods have not identified the white building at the corner of Figure 8.30(b), which appears to be a landmark due to its visual characteristic - colour. This is due to the non-consideration of the colour attribute in this work for the generation of landmarks saliency. The value of the colour attribute cannot be taken as a permanent

measure to retrieve landmark saliency since it can be changed over time due to natural and physical effects. At night, the value of colour does not provide any significance to determine landmark saliency of a building.

Salient landmark results - II: Newham area

The five (5) buildings near Ballam Street and Dongola Road West (see Figure 8.31(a)) identified as salient by the J48 implementation have significant scores by the framework and the method based on the MAD, emphasising the results of the J48 implementation. One of the distinctions in the results is that the method based on the MAD has identified that both visual and structural characteristics of all five (5) buildings have had significance in determining the landmark saliency which the two other methods have not been able to identify. The significance of both characteristics is also emphasised when viewing the map and the Google street view.

The building with IDN 5 at Dongalla Road West is also emphasised to be a strong landmark because of its height as shown in the street view in Figure 8.30(d). According to the street view in Figure 8.31(b), building with IDN 1 appears to be a landmark since it is isolated at the junction (corner). In general, when visually assessed on the street view, the strongest landmark while moving in the direction of A to B is building with IDN 1 while building with IDN 5 becomes the strongest in the direction of B to D. Buildings with IDNs 1 and 5 appear to be strong landmarks when observing the significance scores by the method based on the MAD as well.

Out of all five (5) buildings, the weakest building to be considered as a landmark is the building with IDN 3 when viewing the street views in Figure 8.31. This is more emphasised when observing the significance score of this building assigned by the method based on the MAD.

Salient landmark results - III: Newham area

The building located at the corner to the left of Whitewell Road when moving from A to B is shown in Figure 8.32(a) is a salient landmark based on its size which is a visual

characteristic according to the J48 implementation. The framework has also identified it to be a landmark, indicating more significance of its structural (located at the junction) and semantic characteristics (priority value of 2 as its function is a community centre) as shown in Figure 8.32(d). However, the method based on the MAD has identified the significance of both its visual and structural characteristics compared with the two other methods as emphasised by the street view.

When viewing the street view in Figure 8.32(b), moving from A to B along the Whitewell Road would make no prominent clue to identify building with IDN 1 as a salient landmark other than its smaller height compared to other high-rise buildings. When moving from C to B along Grant Street, it appears to be at the junction (corner) with a smaller height as compared to other buildings in the vicinity when viewing the street view in Figure 8.32(c). Further, it is made salient with its sign board describing its function, appearing on the street view (not very clear), which is one of the semantic characteristics. However, the enrichment of the attributes that describes the function of each building was not part of this research due to the inadequacy of relevant information in the data sets used for deriving landmark saliency.

Salient landmark results - IV: Newham area

The landmark saliency of the two buildings shown in Figure 8.33(a) identified by the J48 implementation on Barking Road has been identified by the framework and the method based on the MAD with their significance scores (see Figure 8.33(c)). When moving in the direction from A to B along the Barking Road, buildings with IDNs 1 and 2 (Figure 8.33(a)) appear salient when viewing the street view in Figure 8.33(b). According to the street view, building with IDN 1 becomes salient with its sign describing its function 'Dental surgery' while building with IDN 2 becomes salient because of its location (situated at the corner) and its shape. The discerning attribute used to identify building with IDN 2 by the J48 implementation is also the same - 'diverse sides' - indicating that it has differently oriented walls. The building with IDN 1 has been identified as salient by the J48 implementation based on the attribute 'orientation to road' which is one of the structural properties. This is also evident from the significance score of the structural property by

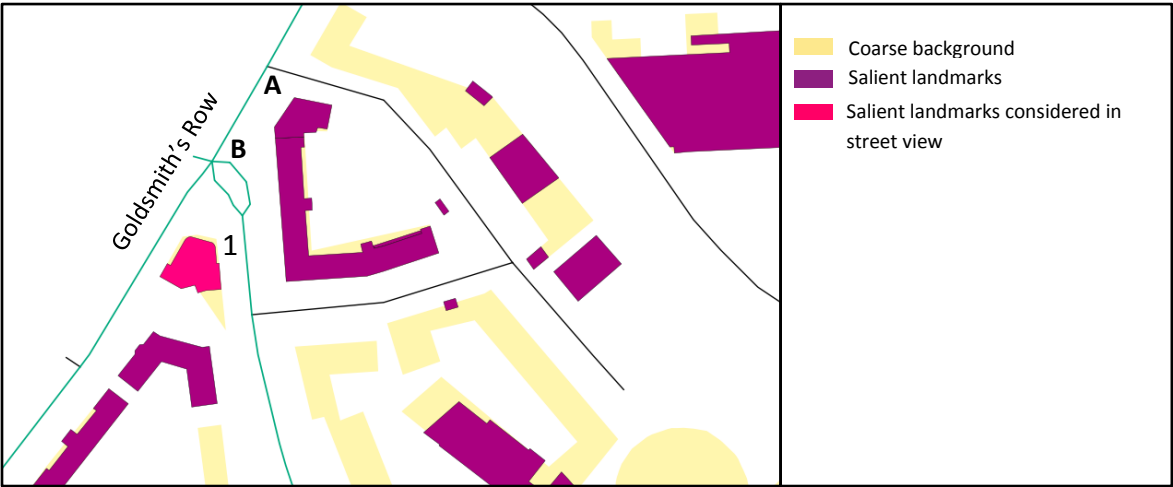
the framework. Out of the two buildings, the most significant building is the one with IDN 2, which is further evident in the street view from the high score given in the framework and the method based on the MAD. The method based on the MAD has identified both buildings as landmarks using both visual and structural characteristics which the two other methods have not been able to identify.

In Figures 8.35 to 8.37, each example is first shown in an inset on the focus map in Figure 8.34 (inset A depicts the map in Figure 8.35, B depicts the map in Figure 8.36 and C depicts the map in Figure 8.37), giving generalized background and salient landmarks at specific locations in Tower Hamlets area, along with the photographs from Google street view depicting the numbered salient landmarks. The calculations of the salient landmarks using the framework of Raubal and Winter (2002) and the new method developed using the MAD are given.



Figure 8.34 Focus map of Tower Hamlets area with the insets A, B and C of the specific locations considered in the validation of the landmark saliency. Note: Map is not printed to scale.

Salient landmark results - I: Tower Hamlets area



(a)



(b)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		$\alpha 1$	$\alpha 2$	$\alpha 3$	ϵ	$\alpha 1$	$\alpha 2$	$\alpha 3$	ϵ
1	Elongation: 0.8	0.33	0.25	0	0.58	3	2	0	1.67

(c)

Figure 8.35 Salient landmark visualization on Goldsmith’s Row: (a) map (b) view towards A to B on the map and (c) calculations of salient landmarks. $\alpha 1$: visual significance, $\alpha 2$: structural significance, $\alpha 3$: semantic significance and ϵ : total significance.

Salient landmark results - II: Tower Hamlets area



(a)



(b)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		$\alpha 1$	$\alpha 2$	$\alpha 3$	ϵ	$\alpha 1$	$\alpha 2$	$\alpha 3$	ϵ
1	Size: 678m ²	0.33	0	0	0.33	4	2	0	2
2	Size: 1001m ²	0.33	0	0	0.33	2	2	0	1.33
3	Importance : 2 (public)	0.17	0.2	1	1.37	2	2	1	1.67

(c)

Figure 8.36 Salient landmark visualisation on St. Peter's Close: (a) map (b) view from A to B on the map and (c) calculations of salient landmarks. $\alpha 1$: visual significance, $\alpha 2$: structural significance, $\alpha 3$: semantic significance and ϵ : total significance.

Salient landmark results - III: Tower Hamlets area



(a)



(b)



(c)

LM IDN	Discerning attribute and its value for landmark saliency with J48	Framework by Raubal and Winter (2002)				Method using the MAD			
		$\alpha1$	$\alpha2$	$\alpha3$	ϵ	$\alpha1$	$\alpha2$	$\alpha3$	ϵ
1	Minimum distance to neighbour:5m	0.33	0	0	0.33	3	3	0	2
2	Number of corners :10	0	0.20	0	0.20	1	2	0	1
3	DEM height : 31m	0.17	0.20	0	0.37	4	1	0	1.67
4	DEM height : 65m	0.5	0	0	0.5	3	3	0	2

(d)

Figure 8.37 Salient landmark visualisation on Centre Street: (a) map (b) view from A to B (c) view from B to A on the map and (d) calculations of salient landmarks. $\alpha1$: visual significance, $\alpha2$: structural significance, $\alpha3$: semantic significance and ϵ : total significance.

Salient landmark results - I: Tower Hamlets area

The landmark saliency of the single building with IDN 1 shown in Figure 8.35(a) identified by the J48 implementation at the junction (corner) on the Goldsmith's Row has also emphasised by the framework and the method based on the MAD with their significance scores in terms of both visual and structural characteristics (see Figure 8.35(c)). When moving in the direction from A to B along the Goldsmith's Row, this building appears salient when viewing the street view in Figure 8.35(b) mainly due to its shape and colour. The discerning attribute used to identify building with IDN 1 by the J48 implementation is the elongation which is one of the criteria to describe its shape (it is a value between 0 and 1). When this value equals to 1, the building is very less elongated.

Salient landmark results - II: Tower Hamlets area

The landmark saliency of the three buildings in the vicinity of St. Peter's Close shown in Figure 8.36(a) as identified by the J48 implementation have been emphasised by the framework of Raubal and Winter (2002) and the method based on the MAD with their significance scores (see Figure 8.36(c)). The J48 implementation has identified building with IDN 3 to be salient with its 'priority' value. The two other buildings with IDNs 2 and 3 are identified as salient due to their size by the J48 implementation. This is further evident when viewing the street view, although they are residential buildings in terms of function (use).

The most salient feature identified by the framework is the building with IDN 3, which is a church with more emphasis on its structural characteristics (see Figure 8.36(c)). The framework has added a high weight to its semantics. This is due to the existence of only one semantic attribute - priority - in the data, which does not have any effect on the averaging of the attribute values of each salient measure derived from this framework as described in Table 8.4. However, when analysing the significance measures given by the new method on the MAD for building with IDN 3, it has assigned equal significance for both visual and structural measures of building with IDN 3 (a church). Its saliency in terms of visual characteristics is further evident when viewing the street view in Figure 8.36(b) in the direction from A to B along St. Peter's Close. The most salient feature identified by the

method based on the MAD is the building with IDN 1 as evident when viewing the Google street view due to its size and height, although it is a residential building.

Salient landmark results - III: Tower Hamlets area

The landmark saliency of the four (04) buildings visible from Centre Street shown in Figure 8.37(a) as identified by the J48 implementation have been emphasised by the framework with its significance scores (see Figure 8.37(c)). However, the method based on the MAD has identified only buildings with IDNs 1, 3 and 4 as salient. It has not identified building with IDN 2 as salient due to its fewer visual characteristics. The most significant salient feature identified by the framework and the method based on the MAD is building with IDN 4 which appears to be a global landmark due to its height when viewing the street view in Figure 8.37(c), moving along the road from B to A. The J48 implementation too has identified it with its discerning attribute - height. Building with IDN 3 too is identified as a landmark due to its height by the J48 implementation, although it is not apparent to be an immediate local landmark on the road. Building with IDN 2 is identified as salient by the J48 implementation based on the number of corners of a building, which is a visual characteristic while the framework has identified it to be salient due to its structural characteristics (this building is located at the corner). In this situation, the structural characteristic - corner - appears to be more prominent than the visual characteristic, number of corners. However, when viewing the street view, it can be identified as salient because of both characteristics - visual (height is very low compared to other buildings in the vicinity) and structural (its location at the corner). The residential building with IDN 1 has been identified as a salient landmark by the J48 implementation, considering its minimum distance to neighbouring buildings, which is one of the structural characterises while the framework has identified it as a landmark due to its shape (its shape is non-orthogonal in the 'orthogonal' attribute). However, when viewing this building in the street view, it becomes salient due to both visual and structural characteristics. It has been identified by the method based on the MAD as one of the two most salient landmarks. Further, the method based on the MAD has identified both visual and

structural characteristics of all the four buildings to be significant while the two other methods have not been able to identify both of those characteristics of each building.

All long buildings appear as salient landmarks to the right of the road when moving from A to B in Figure 8.37(a) due to their size and height (the method based on the MAD has only identified a few of those buildings as landmarks - see the focus map derived from this method in Figure 8.44, page 322). These buildings do not appear to be landmarks on the ground because they all look the same in shape and height (see Figure 8.38 to visualise part of such buildings from the middle of the Centre Street). Collectively, they are very noticeable as a group, which would help a wayfinder to orient in the right direction. However, there are limitations in the automatic derivation of landmarks, especially in areas where all the buildings are high-rise buildings/structures with the same height.



Figure 8.38 Google street view of high-rise buildings with the same shape and the height visualized from the middle of the Centre Street.

Validation of landmark saliency on a decision point – Tower Hamlets area

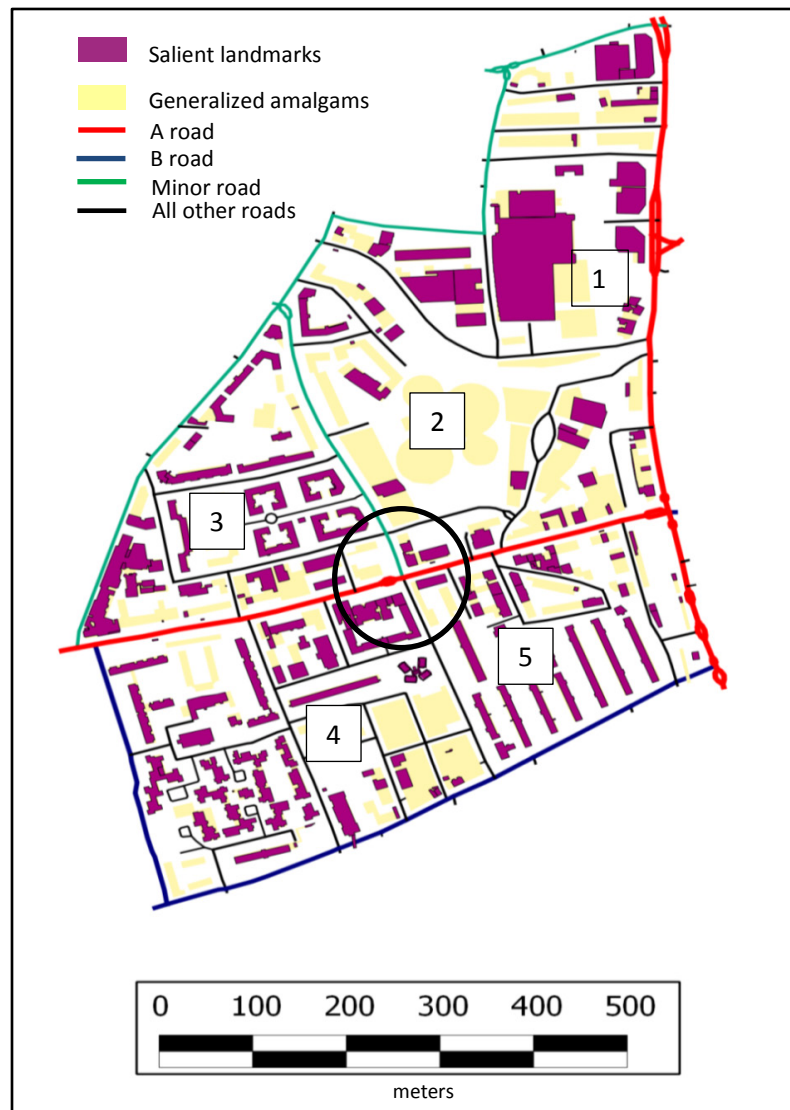


Figure 8.39 Focus map of Tower Hamlets area with the decision point circled, and the regions (1 to 5) used in deriving landmark saliency. Note: Map is not printed to scale.

Figure 8.39 depicts the decision point (circled in black colour) used to validate landmark saliency. The selection of this decision point is due to it being in a corner of the four intersecting regions - 2, 3, 4 and 5 - used in deriving landmark saliency with the J48 implementation of the focus map above in Figure 8.39. The J48 implementation is applied to each of these regions separately by dealing with sensitivity analysis and attribute discretization (transformation) based on the characteristics of data. Therefore, it is important to see how the J48 implementation derives landmark saliency of buildings

around a buffer of 50m from the junction (decision point), falling within all of these four regions.

When the J48 implementation is applied to the buildings chosen within a radius of 50 m from the decision point (see Figure 8.40(a)) without attribute discretization (transformation), none of the buildings are selected as salient at the topmost level (zero level) of the decision tree because the J48 implementation of the C4.5 algorithm has not detected any of the outliers based on the gain ratio as discussed in Section 2.5.1, page 55. However, at the first level of the tree, 19 buildings are identified as salient though it does not give any prominent information at the decision point. When the attributes - DEM height, size and the minimum distance to road - are discretized, it identifies four buildings shown highlighted in Figure 8.40(b) to be salient at the zero level of the decision tree. Highlighted values in Table 8.11 are the attribute values, the J48 implementation chooses to make each of the four landmarks salient. When comparing the salient landmarks derived in regions with that of derived at the junction around a buffer, it can be seen that the results are different (see Figure 8.40). Building with IDN 20 identified as salient at the decision point has not been identified as a landmark when the J48 implementation has derived landmarks in region 3. This further emphasises that the application of the J48 implementation is region dependent.

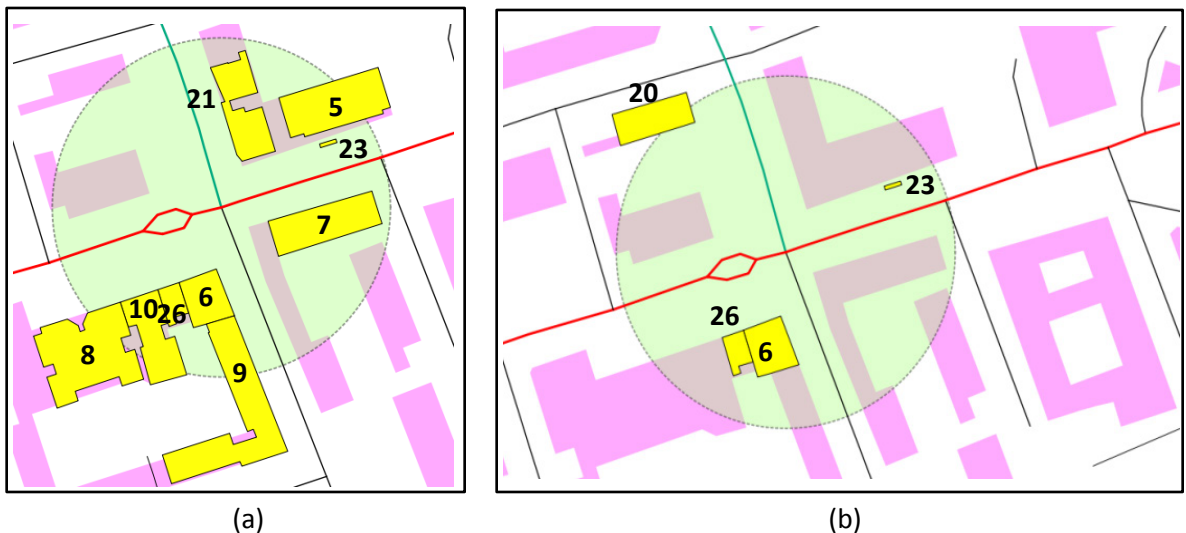
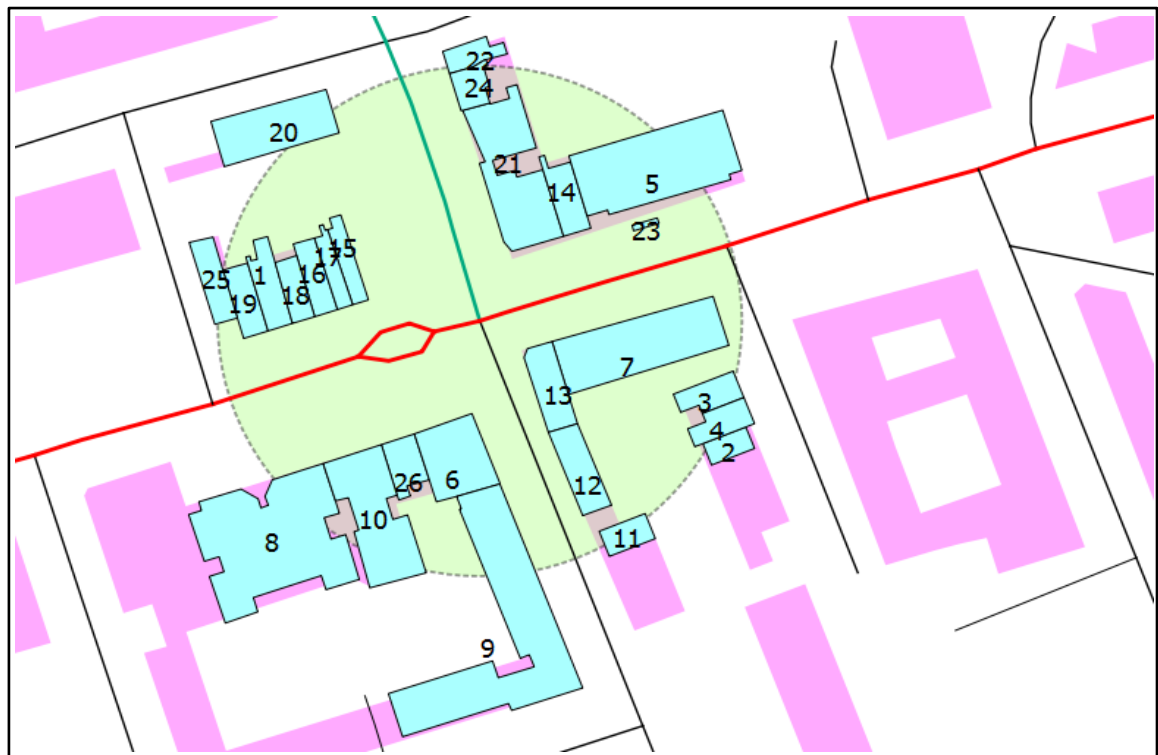
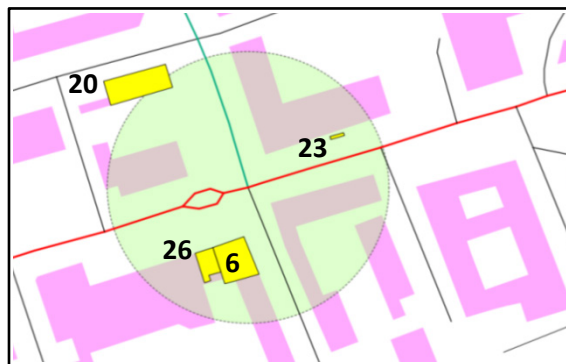


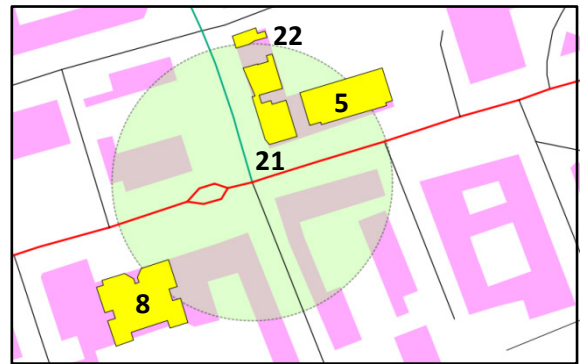
Figure 8.40 Evaluation of salient landmarks at a decision point with the J48 implementation: (a) Salient landmarks falling within the buffer of 50m, derived from each region separately (regions 2, 3, 4 and 5 in Figure 8.39 and (b) salient landmarks derived from all the buildings within the same buffer at the decision point during focus map generation in Tower Hamlets area.



(a)



(b)



(c)

Figure 8.41 Evaluation of the landmark saliency at a decision point in Tower Hamlets area: (a) all the buildings around the decision point within a radius of 50m (b) salient landmarks chosen by the J48 implementation and (c) salient landmarks chosen by the implementation of the framework of Raubal and Winter (2002) from the buildings with a total significance ≥ 0.75

When observing the total significance (ϵ) of these four buildings identified by the J48 implementation as landmarks in Figure 8.41 (b), each building has a significant value > 0 according to the results given in Table 8.11. This indicates that these four buildings are potential landmarks at the decision point, although discerning attributes chosen provide

no clear clue about their salience. However, all these buildings are in the residential category with a less building importance (a 'priority' value of 5).

Table 8.11 Landmark significance together with attributes and their values of the four chosen salient landmarks with the J48 implementation.

IDN	α_1	α_2	α_3	α_4	α_5	α_6	β_1	β_2	β_3	β_4	β_5	β_6	β_7	Υ_1	σ_1	σ_2	σ_3	ϵ
26	29	67	6	0.6	1	0	15.1	2	342	corner	1	0	0.0032	5	0.33	0	0	0.33
23	17	6	4	0.2	1	0	8	0	72	corner	0	3.2	0.0031	5	0.33	0.14	0	0.48
6	31	175	4	0.8	1	0	8	2	342	corner	1	0	0.0036	5	0.33	0	0	0.33
20	13	207	4	0.4	1	0	6	1	73	corner	1	0	0.0037	5	0.33	0	0	0.33

α represents visual properties: α_1 – DEM height, α_2 – size, α_3 – no. of corners, α_4 – elongation, α_5 – orthogonal, and α_6 – diverse sides.

β represents structural properties: β_1 – minimum distance to road, β_2 – no. of adjacent neighbours, β_3 – orientation to North; β_4 – orientation to road, β_5 – average orientation to neighbours, β_6 – minimum distance to neighbour, and β_7 – neighbourhood density.

Υ represents semantic properties: Υ_1 – Importance.

σ_1 – Visual significance, σ_2 – Structural significance, σ_3 – Semantic significance, and ϵ – Total significance.

Table 8.12 Landmark significance together with attributes and their values of the four chosen salient landmarks with a total significance ≥ 0.75 based on the framework by Raubal and Winter (2002).

IDN	α_1	α_2	α_3	α_4	α_5	α_6	β_1	β_2	β_3	β_4	β_5	β_6	β_7	Υ_1	σ_1	σ_2	σ_3	ϵ
5	29	404	8	0.4	1	0	12.1	1	72	corner	0	0	0.0031	1	0.33	0	1	1.33
8	34	570	24	0.9	1	0	17.5	4	72	corner	1	0	0.0033	3	0.67	0.14	0	0.81
21	23	290	17	0.4	1	0	8.5	5	343	corner	0	0	0.0028	1	0.33	0.14	1	1.48
22	25	47	8	0.4	1	0	4.7	1	73	corner	0	0	0.0027	1	0.33	0	1	1.33

α represents visual properties: α_1 – DEM height, α_2 – size, α_3 – no. of corners, α_4 – elongation, α_5 – orthogonal, and α_6 – diverse sides.

β represents structural properties: β_1 – minimum distance to road, β_2 – no. of adjacent neighbours, β_3 – orientation to North, β_4 – orientation to road, β_5 – average orientation to neighbours, β_6 – minimum distance to neighbour, and β_7 – neighbourhood density.

Υ represents semantic properties: Υ_1 – Importance.

σ_1 – Visual significance, σ_2 – Structural significance, σ_3 – Semantic significance, and ϵ – Total significance..

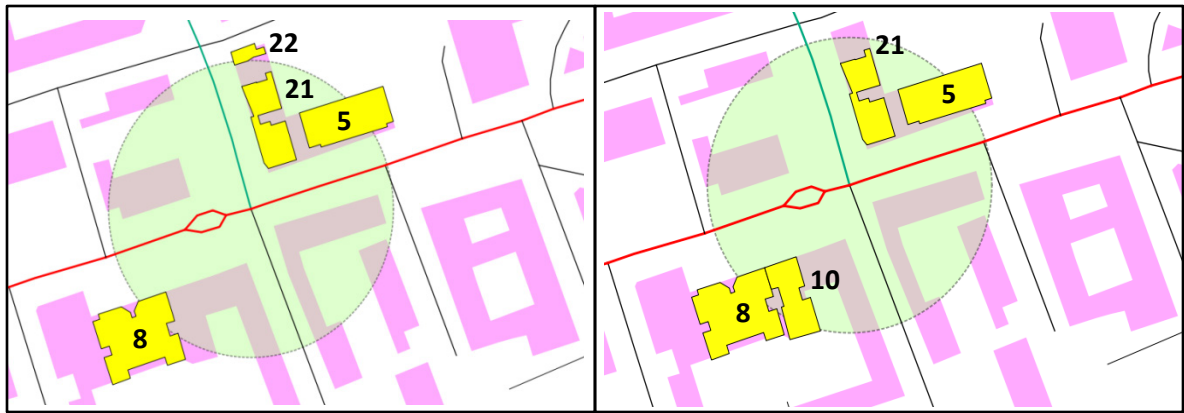
When the implementation of the framework for deriving landmark saliency on these buildings is applied at the decision point with equal weight assigned to each of the three characteristics, it gives different results to that of the J48 implementation. Table 8.12 lists the attribute values and the corresponding significance values of the four buildings with a total significance > 0.75 . Figure 8.41(c) depicts these four buildings overlaid on the coarse background. The most salient landmark is the building with IDN 21 with a total significance

of 1.48 according to the Table 8.12. It is a wholesale store (attribute Name: G B Wholesale Ltd). When investigating the function of the other three buildings; building with IDN 5 is a car sales (attribute Name: Lees cars), building with IDN 22 is a café (attribute Name: Billy's Café) and building with IDN 8 is a property sale company (attribute name: Columbia Estates Ltd). However, all these four buildings seem to be important landmarks at this decision point. The selection of landmark saliency at a decision point can be a group of buildings based on a criterion using higher total significance values of buildings rather than selecting the building with the highest significance score. When comparing the results of landmark saliency with the results obtained from the J48 implementation at the decision point, the framework of Raubal and Winter (2002) gives promising results excluding all the four buildings chosen as salient landmarks by the J48 implementation.

Comparing the method using the MAD developed in this research

When further validating the results of the J48 implementation with the results obtained using the method based on the MAD developed in this research, the new method has derived potential landmarks at the decision point. The framework of Raubal and Winter (2002) has identified building with IDN 21 as the most salient landmark at the decision point when observing the significance scores (see Tables 8.12 and 8.13) while the new method has identified two buildings with IDNs 21 and 8 as the most salient landmarks. These two buildings serve as the most salient features when observing the three measures of visual, structural and semantic characteristics.

When observing the structural significance of buildings with IDNs 5, 10 and 22 (see Table 8.13) given by the framework and the new method, it is evident that the framework has not detected their structural measure (orientation to the road - corner) while the new framework has identified this measure. The reason is that the ordinal values assigned to the attribute 'orientation to road', considered in the statistical decision criterion used in this framework, have not become sensitive in outlier detection. In the calculation of the significance measure of the attribute 'orientation to road', the significance has been directly assigned a binary significance value (see Table 8.7, page 287) without considering the statistical decision criterion by the new method based on the MAD.



(a)

(b)

Figure 8.42 Evaluation of the landmark saliency at a decision point in Tower Hamlets area with all the buildings around the decision point within a radius of 50m: (a) salient landmarks chosen by the framework of Raubal and Winter (2002) from the buildings with a total significance ≥ 0.75 and (b) salient landmarks chosen by the new method on the MAD from the buildings with a total significance ≥ 1 and the significance contribution from two or more significance measures (see Appendix F.7 for the attributes and the significance scores of each building considered at the decision point with the new method on the MAD).

Table 8.13 Comparison of landmark significance together with individual significance scores of each measure of the most prominent salient landmarks chosen based on the framework by Raubal and Winter (2002) and the new method based on the MAD.

IDN	Framework by Raubal and Winter(2002)				Method using on the MAD by this research			
	$\alpha 1$	$\alpha 2$	$\alpha 3$	$\epsilon 1$	$\beta 1$	$\beta 2$	$\beta 3$	$\epsilon 2$
5	0.33	0	1	1.33	1 (size)	1 (orientation to road)	1 (priority)	1
8	0.67	0.14	0	0.81	2 (size / no. of corners)	1 (orientation to road)	1 (priority)	1.33
10	0.33	0	0	0.33	1 (size)	1 (orientation to road)	1 (priority)	1
21	0.33	0.14	1	1.48	2 (size / no. of corners)	1 (orientation to road)	1 (priority)	1.33
22	0.33	0	1	1.33	0	1(orientation to road)	1 (priority)	0.67

$\alpha 1, \beta 1$ – visual properties, $\alpha 2, \beta 2$ – structural properties, $\alpha 3, \beta 3$ – semantic properties and $\epsilon 1, \epsilon 2$ – total significance.

When further observing the significance measures of the building with IDN 5 in Table 8.13, given by the framework, its semantic measure has got a higher significance. The reason is that the semantic measure has only one attribute - ‘priority’ - in detecting landmark

saliency. Hence, when averaging the significance score of the semantic measure of the building with IDN 5 with the number of attributes according to the framework, the significance score remains the same (i.e. value - 1). When averaging the scores of two other measures, they have a higher number of attributes (visual - 6 attributes and structural - 7 attributes). As a result, they tend to get lower significance scores (i.e. values < 1). Therefore, the total significance score of this building in the framework has given more emphasis due to the semantic measure. However, in calculating the individual score in the method based on the MAD, averaging is not considered (see Table 8.8, page 288). As a result, the building with IDN 5 has not become more emphasised as a salient feature based on the new method, although this is not the only reason for the lack of emphasis.

8.3 Discussion

This research has been able to identify issues in spatial data structuring with Delaunay triangulation, data enrichment for spatial clustering and cluster shape enrichment for subsequent automatic generalization process, and extraction of required attributes for deriving salient landmarks. The salient landmarks are identified using the J48 implementation of the C4.5 decision tree algorithm under data mining together with automatic map generalization of building polygon data to generate coarse background on focus maps. These fill the gaps in understanding the processes of generating focus maps for wayfinding.

In generating the results, the first step is the creation of building clusters for the subsequent building generalization process to create a coarse background of building geometry to form the basis for generating focus maps automatically. The spatial clustering algorithm developed in this research can create building polygon clusters based on the hierarchical application of the Gestalt factors - proximity, orientation and similarity in shape. In this process, more emphasis is given to the context of buildings such as the road network and the hydrographic features in regions delineated by the user in order to avoid clustering of buildings across contextual features so as to reduce complexity of data handling and to improve the efficiency of data processing. However, when applied to the

test data sets, the triangulation algorithm gave an exception, terminating the automatic clustering process. A fix was made to rectify this issue as explained in Section 8.1.1.

Qi and Li (2008) have carried out the hierarchical clustering on building features using the Gestalt factors - proximity with the CDT, orientation with the MBR, and similarity with the overlap ratio between a pair of buildings. The CDT used by them does not create explicit adjacency relationships between buildings, depending on the configuration of shape and position of buildings as discussed by Ware and Jones (1996) and Ai *et al.* (2007), although most of the researchers have used this triangulation structure for building clustering as discussed in Section 2.4.5, page 52. Further, the application of the MBR for deriving orientation is not a satisfactory measure when the shape of the building is square or terraced (Duchêne *et al.*, 2003). However, the MBR is consistent with buildings of irregular shape. Qi and Li (2008) have initially partitioned buildings into regions using contextual features, applying a clustering process to the buildings within a particular region, ignoring contextual features existing in a region such as hanging roads (roads with dead ends). If all or some of the inner contextual features are to be retained at the target scale, reclustering of buildings is required in instances where such inner contextual features cross the generated clusters. The most important fact is that they have not tested the results of their clustering approach. The hierarchical clustering process developed in this research has provided solutions to all these shortcomings in the approach of Qi and Li (2008). The research has developed an improved constrained triangulation algorithm termed as DCT to deal with implicit neighbourhood relations caused by the CDT in the proximity calculation between building polygons. The statistical wall orientation developed by Duchêne *et al.* (2003) has been used for the calculation of the orientation difference between each pair of buildings, since the MBR cannot deal with buildings of square or terraced shape (see Section 5.2.1, page 115). However, the statistical wall orientation is not consistent with circular or irregular shaped buildings according to Duchêne *et al.* (2003). They have suggested using a combination of the MBR and the wall statistical weighting for deriving orientation. However, this does not provide consistent results in the case of clustering in this work. Therefore, the wall statistical weighting is used since circular or irregular shaped buildings occur very rarely in the data sets. The

wall statistical weighting gives promising results in cluster generation in the test data sets (except the circular building like structures existing in region 2 depicted in Figure 8.13 in Tower Hamlets test data where an arbitrary value is assigned to all circular shaped buildings in the clustering process). It is important to note that an intelligent cluster classification system has been developed based on the three Gestalt factors and the MST segmentation in this research. Based on the internal evaluation of the results of the visual perception test in the generated clusters, carried out using the secondary data obtained from the NMA of Sri Lanka as explained in Section 5.3.3, page 133, it is identified that the incorporation of contextual features is necessary to avoid cluster formation across these features. Thus, the developed DCT algorithm has been replaced with the CNDT algorithm by Ruppert (1995) described in Section 4.2, page 85 since it cannot presently deal with linear features. This algorithm has been further customised to apply constraints on building edges and deal with Steiner points. The other advantage of using the CNDT is that it creates Steiner points to make the triangulation Delaunay stable where all explicit topological links between buildings and contextual features are captured. These links are necessary to the clustering process as well as for the extraction of attributes required to mining landmark saliency. That these Steiner points are not created physically on the data is another advantage of keeping the original data intact during the triangulation process. The improvement of the clustering algorithm to deal with contextual features has made it very convenient to partition the regions without excluding inner contextual features such as roads with dead ends (only roads are available in the test data sets for the context).

Only Regnauld and Revell (2007) have taken into consideration roads as contextual features in generating clusters with the proximity graphs out of all the triangulation approaches for building clustering discussed in the literature in Section 2.3.3, page 35. They have used the CDT with splitting the road network at regular intervals (densification of points) in order to ensure each building is hooked to a nearby road. However, in this approach the topological relations between buildings can be lost since the CDT can create implicit topological relations as discussed above. The other issue is that there is no guarantee that all the buildings are linked to the closest road segment in the densification process of the road network, depending on the distance applied for densification. Such

densification affects the original data by adding new vertices in the data set as well. Further, the clustering approach does not generate intelligent knowledge about each cluster created with a cluster classification system using the Gestalt factors. They have only considered proximity in the approach.

A further enrichment in terms of cluster shape and its orientation is carried out after the clustering process based on the statistical wall orientation algorithm by Duchêne *et al.* (2003) to select the choice of aggregation generalization operations developed to deal with clusters with the orthogonal and the non-orthogonal shapes defined as described in Section 5.4.2, page 154. This enables the ability to give an enhanced visualisation and meaning to the building amalgams (orthogonality of edges are retained as much as possible in the aggregation and narrow corridors and juts are exaggerated) after aggregation as depicted in the generalized results in Section 8.2 and in the validation results in Section 8.2.2.

The second step after clustering and shape enhancement is to extract all the geometrical and the structural information given in Table 6.1, page 166 automatically from the two topographical data sets used as the test areas and store them in the PostgreSQL database coupled with PostGIS extension to handle spatial data. The only attribute considered to enrich thematic values (semantics) of a building is the building priority ranking measure assigned in terms of an ordinal value based on the building function and use as given in Table 6.2, page 179. Also, methods and algorithms have been developed for extracting geometric (visual) and spatial (structural) attribute values based on the topological information between the building geometries and the contextual data using the CNDT customised for the building clustering process. A fix was required to deal with null pointer values in the algorithm developed to derive building orientation in relation to the closest road segment when applying to the test data set of Newham area as mentioned in Section 8.1.2. It should be noted that although this algorithm derived results automatically, some incorrect assignment of orientation values occurred to buildings located at corners as already discussed in Section 8.1.2. Such unexpected results had to be investigated and rectified interactively before proceeding to the next step of deriving landmark saliency.

However, there were a very few such exceptions that had to be dealt with. Further work would be necessary to make it completely automated.

The third step is the application of the J48 implementation of the C4.5 decision tree algorithm by Quinlan (1993) on the test data sets that have been enriched with geometrical (visual), spatial (structural) and thematic (semantic) attributes for the identification of salient landmarks. According to the literature, the ID3 decision tree algorithm and the COBWEB clustering have already been tested to drive landmark saliency at decision points (Elias, 2003; Elias, Hampe and Sester, 2005). The J48 implementation of the C4.5 decision tree supervised classification method is used in this work after comparing and evaluating its results together with the results obtained from the ID3 and the COBWEB implementations as described in Section 6.2.4, page 195 using the open source WEKA Java library. This testing and evaluation is carried out at a decision point as well as in a region delineated by the road network. For this purpose, the three methods are customised to traverse through the levels of the tree to capture salient landmarks with Java object-oriented programming language.

The test data in Tower Hamlet area are very complex, comprising of a variety of shapes of building geometries with small and narrow attachments including structures which are identified as single entities with all the attributes. Some of these small and/or narrow attachments have very discerning heights either much lower or higher than the surrounding buildings. These buildings tend to stand out as salient landmarks during the landmark derivation process. Thus, buildings with such exceptions are pre-processed before the application of the salient landmark derivation process. For example, if a small building attached to the main building has a height greater than the main building, the height of the small building is assigned to the main building (in another attribute column, keeping the original height of the building intact) and the small building is removed from the salient landmark derivation process. However, before removing such a building from the derivation process in a particular region, its building priority value is checked with that of the main building. If the priority is higher than the main building, it is not removed, and even the height value of the main building is not changed.

There were some buildings in the data sets that did not have any orientation or the minimum distance to a road (null values) in relation to the road network after data enrichment. The reason was that these were all in between other buildings, thereby not having a direct link to a nearby road within a region. Such buildings were removed from the salient landmark derivation process since they too tended to stand out as salient landmarks when the dummy values were assigned during the landmark derivation process.

The visibility of a building is an important visual property in determining landmark saliency. In order to derive a visibility factor for a building, topography in the vicinity of the building, its height and observer's height and position (viewpoint) need to be taken into account. Work on visibility has already been carried out by Brenner and Elias (2003) and Nothegger, Winter and Raubal (2004). However, the visibility check has not been performed in the two test areas since they have almost flat topography, hence not incorporated in deriving landmark saliency. The observer's viewpoint is also an important visual measure in a wayfinding application where any position on a road is potentially a viewpoint. Especially in pedestrian wayfinding, this is the public street area. Raubal and Winter (2002) have defined this in terms of 2D visibility. However, this is not included in the attribute derivation under data enrichment, considering the time involved in developing another algorithm for this purpose. With the building links and their corresponding road network links, a measure for each building on 2D visibility can be developed.

When the results of salient landmarks obtained using the J48 implementation are analysed with those obtained using the framework to derive landmark saliency by Raubal and Winter (2002) and the new method developed in this research based on the MAD, it is identified that the J48 implementation provides rather satisfactory results when applying to regions (see Tables 8.5 and 8.6 for the comparison of the J48 implementation results with the framework of Raubal and Winter (2002), and Tables 8.9 and 8.10 for the comparison of the same with the method based on the MAD in the previous section). This is further emphasised by the visualisation of salient buildings with the Google street view

in chosen cases under the validation of salient landmarks in the previous section. The attribute discretization plays an important part in determining the quality of results, even though, the J48 implementation can deal with both nominal and numeric data. Also, it is necessary to partition data into meaningful regions such that a region does not contain a large number of buildings. A region with a large number of buildings may lead to making discretization of the attributes difficult to achieve satisfactory results as experienced in region 6 in the test data set of Newham area.

The evaluation of the J48 implementation at a particular decision point which includes a few important landmarks in the Tower Hamlet area within a radius of 50m reveals that in order to get the results, attribute discretization is necessary, even though the data set is very small (only 26 buildings around the decision point as depicted in Figure 8.41 in Section 8.2.3). The salient results are given in this instance at the topmost level (0 level) in the decision tree. However, when analysing these results with those obtained from the framework by Raubal and Winter (2002) and the new method based on the MAD, the results are entirely different. The framework and the method based on the MAD identify most important landmarks (see buildings with IDNs 5, 8 and 21 in Figure 8.42 in Section 8.2.3) at the junction (decision point) while the J48 implementation cannot identify them. The results of significance measures of the new method based on the MAD prove to reflect all the three characteristics (visual, structural and semantic) of a landmark while the framework cannot identify structural characteristics of some salient landmarks at the junction (see Table 8.13 in Section 8.2.3).

Further, when analysing the discerning attributes the J48 implementation has chosen to define a particular building as a landmark, comparing with the results from the method based on the MAD and the Google street view, it is hard to identify a direct clue in some instances as to why some attributes are made discerning. Thus, the outcome of the validation of the results of the J48 implementation has shown that the J48 implementation needs further to be improved or even abandoned in favour of an approach using the MAD to identify salient buildings on topographic data sets, although the ID3 decision tree, which is the predecessor of the J48 implementation (J48 can be

treated as the improved version of ID3), has already been employed by Elias (2003) and Elias, Hampe and Sester (2005) for derivation of landmark saliency in wayfinding. However, they have not conducted any usability test in order to validate the results obtained from the ID3 decision tree. The implementation of the framework developed by Raubal and Winter (2002) can identify landmark saliency in a region and at a decision point where it does not often reflect the significance of visual, structural and semantic measures in order to provide a reliable significance score to detect landmark saliency. The reason is that its decision criterion based on the mean and the standard deviation is impacted by the presence of outliers. The saliency measure proposed by Nothegger, Winter and Raubal (2004) using the MAD provides some scores with a value of infinity for buildings (it can give division by zero in their equation used to calculate the significance score) when applying in this research as discussed in Section 8.2.3.

The new method developed to detect landmark saliency provided improved results when applied to regions and decision points in this research. In order to detect outliers, only the upper limit of the decision criterion (see equation 7 in Section 8.2.3, page 285) was considered in order to avoid selecting buildings with small values of attributes (especially the size) of a building as outliers. However, one of the limitations of this approach was that it eliminated the significance of buildings with low heights. In some instances, a building with a low height at a decision point with all high-rise buildings around would stand out as a landmark (see street view map in landmark results – III: Tower Hamlets area in Section 8.2.3, page 303 for such a building (building with IDN 2)). One of the solutions was to classify the size of buildings into a new range of values in a meaningful way so as to clearly distinguish the buildings visually using data transformation and apply a decision criterion for both limits. This approach would add significance to buildings with low heights as well. It would need further testing with different data sets in future research. Figures 8.43 and 8.44 depict the focus maps reproduced region based with the method based on the MAD in Newham and Tower Hamlets areas respectively for the validation of landmark saliency derived from the J48 implementation as described in Section 8.2.3.

The Focus map produced with the method MAD – Newham area

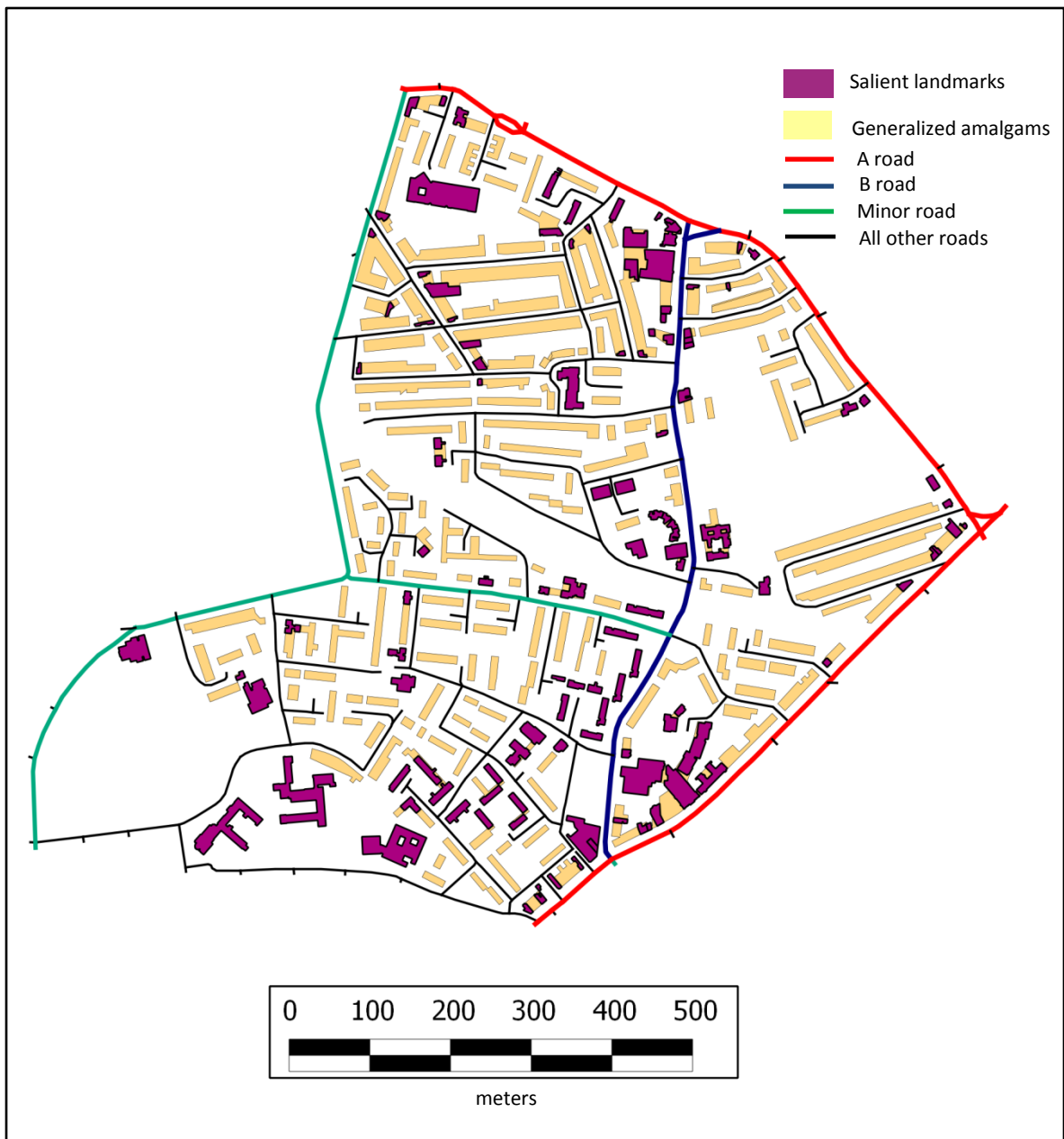


Figure 8.43 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Newham area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

The Focus map produced with the method MAD – Tower Hamlets area

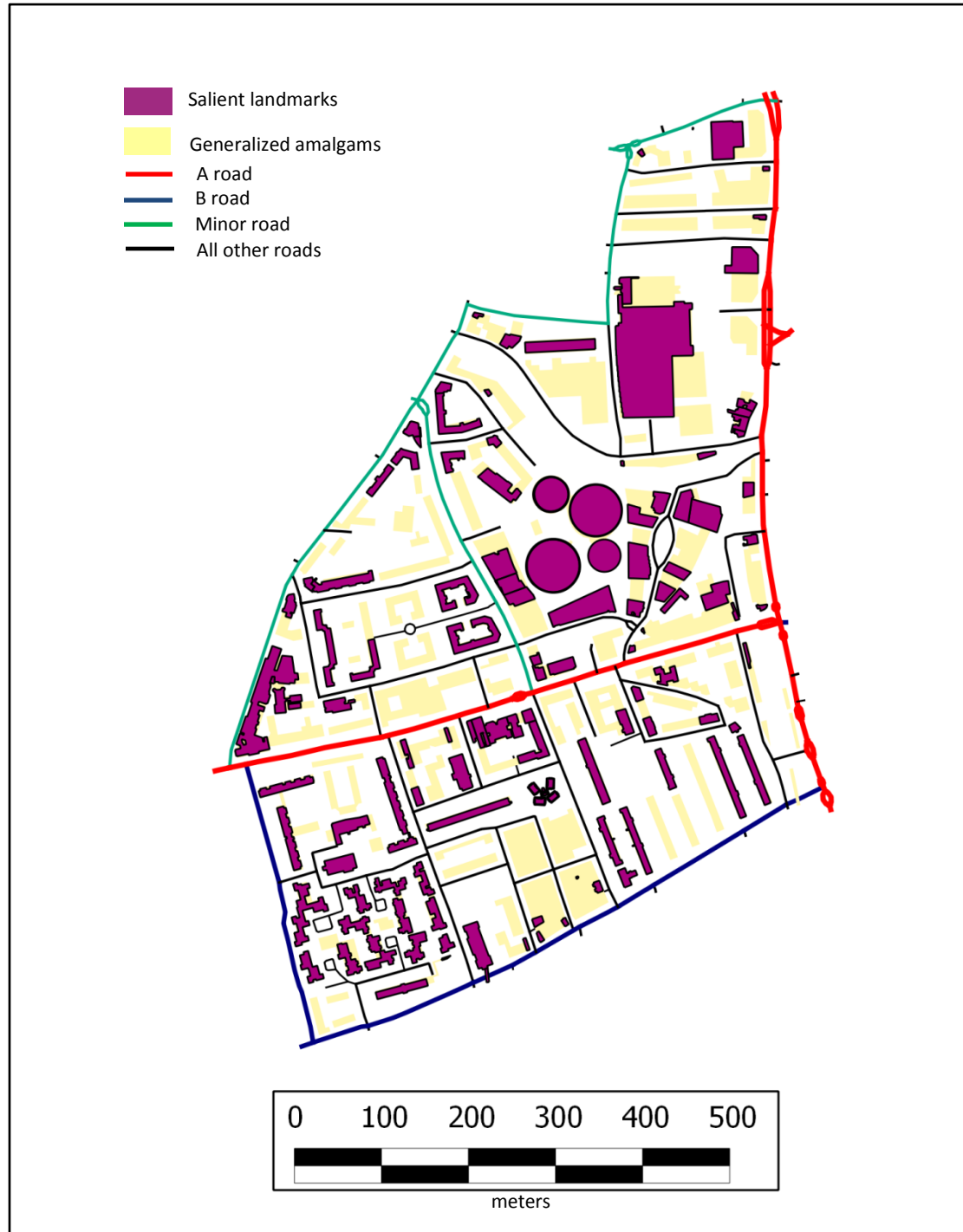


Figure 8.44 Focus map with salient building landmarks highlighted with graphical variable - colour - portrayed in the original shape on the coarse background of the generalized amalgams of buildings at the target scale of 1 : 8K, derived from the source data at the scale of 1 : 1.25K in Tower Hamlets area (part of). Note: Map is not printed to scale. Data source: OS MasterMap, Crown copyright.

The fourth and the final step is the application of the automatic generalization on enriched building clusters in the two test areas to generate coarse background in order to support and enhance the visualisation of salient landmarks on the focus maps using the generalization algorithms developed and/or modified in this research. This method of generalization of features is called the object transformation technique according to Bereuter and Weibel (2010) as described in Section 2.1, page 13.

The main generalization operation considered in this research is the aggregation operation to create amalgams of buildings with coarse details. This operation is required when a group of buildings appears merged on the target scale of a data set due to the scale reduction in the application of the map generalization. However, after the creation of amalgams at the generalized scale, the two other generalization operations applied are simplification and exaggeration. The simplification is used to remove the edges of buildings that are hardly visible in the target scale. The exaggeration is performed to eliminate narrow juts and/or corridors of the amalgam created as a result of building aggregation applied to a cluster. More emphasis is given to deal with existing issues in creating amalgams from building clusters, identified from the literature review in this research. Two types of aggregation algorithms have been developed in this research: (a) building aggregation with orthogonal sides and (b) building aggregation with non-orthogonal sides where the former is to aggregate cluster of buildings with orthogonal sides and similar orientation and the latter is to deal mainly with clusters of buildings with non-orthogonal edges/and or dissimilar orientations in the cluster outline (see generalization specification used in this work given in Section 8.2.1, page 265). Both algorithms were revised during the results generation process using the test data sets to fix the bugs encountered as explained in Section 8.1.3.

The aggregation algorithm with orthogonal sides can handle buildings at any exceptional position as illustrated in Figures 7.4 and 7.5 in Section 7.2.1, pages 203 and 204. It has the ability to fill the gaps between buildings with buffer operation and create a single amalgam. If the output of buffer operation is not a single amalgam (this happens when buildings are located at exceptional positions), adjacent pairs of polygons are tracked first

with the DCT developed in this research and then subjected to the CNDT with constraint edges between each pair of buildings as described in Section 7.2.8, page 224 in order to create a single amalgam. Then, squaring the edges is performed with fusing the amalgam using the ear polygons followed by exaggerating the juts and the narrow corridors and finally simplifying the edges as described in Sections 7.2.5 and 7.2.6, pages 210 to 217. The final algorithm is the revised version of a series of algorithms subject to testing, evaluation of the results, and the refinement of the algorithms in the development process. Regnauld and Revell (2007) have developed a similar type of algorithm to deal with the exceptional positions of buildings in the clusters with squaring and bridging the gaps by identifying the adjacent pairs of buildings using the MST as discussed in Section 2.2.8, page 29. One of the drawbacks of their algorithm compared to the one developed in this research is that it tends to simplify each building of the cluster with its MBR after orienting the cluster along the x-y axes with its GOBR described in Section 7.1, page 200. This simplification produces overgeneralized squared amalgams. The other drawback identified is that their algorithm relies on the use of the MST to find the adjacent pairs of buildings. This may lead to a loss of some important adjacent relations between the buildings. The ideal solution would have been to use triangulation coupled with the MST. Further, the algorithm cannot deal with exaggerating narrow corridors of the generalized amalgams.

The aggregation algorithm with non-orthogonal sides can also handle buildings at any exceptional position as described above. The algorithm first applies the dilation and erosion operation on the building cluster to create an initial amalgam. If the initial amalgam is not a single polygon, the algorithm first identifies adjacent pairs of polygons using the DCT and then re-triangulates each adjacent pair with the same triangulation and identifies all the triangles between each pair of buildings. Then all these adjacent triangles are spatially sorted to capture adjacent triangles to each single triangle. The gap between each pair of buildings is bridged with the triangles based on the space triangle edge distance threshold (see Section 7.3.1, page 230 for the algorithm) using the pairing algorithm developed in this research. Finally, using the OpenCarto simplification algorithm described in Section 7.3.2, smaller edges of the amalgam are removed based on the distance tolerance (see Section 8.2.1 for specification for simplification).

In the application of the simplification algorithm in the amalgams created using the two test data sets, the algorithm produced some exceptions in simplifying the edges, and a fix was made as described in Section 8.1.4.

The same application of aggregating natural objects (e.g. buildings with irregular shape) by initially finding triangles between a pair of objects and further selecting the final candidate triangles for filling by identifying bridging edges of the initial triangles subject to a distance tolerance with Delaunay triangulation has been described by DeLucia and Black (1987). Based on their technique of choosing candidate triangles to fill the gap between natural objects on a distance threshold, further work has been carried out by Jones, Bundy and Ware (1995) and Ware *et al.* (1995) using the CDT. This approach can leave holes (even triangular holes can be created) between a pair of objects, depending on the shape of the bridging space since no adjacent information of triangles has been taken into account. Since their algorithm considers triangles between a pair of buildings, if 'false' triangles (see Figure 4.11 in Section 4.4.2) exist in the space region between the two objects, they appear as triangular holes in the final amalgam (Figure 7.21 in Section 7.3.1 depicts such an instance). All these issues have been resolved when developing the aggregation algorithm to deal with clusters of non-orthogonal shape in this research. It can either retain or remove inner holes using an area threshold. Further, the initial dilation and erosion algorithm applied to the cluster preserves orthogonal sides if available in the cluster outline. Thus, if all buildings in a cluster are orthogonal and oriented in the same direction, the application of this algorithm can also create an amalgam, preserving orthogonal sides of the original buildings only if the initial dilation and erosion algorithm creates a single amalgam.

It should be noted that if the cluster has all or some attached buildings in the application of both algorithms to a cluster, they are initially fused before applying the aggregation algorithm. If the area of inner holes is less than a particular threshold, such holes are removed. Further, if the GOBR of a cluster is smaller than a given edge distance threshold based on the target scale, such clusters are symbolized using the algorithm developed in this research as described in Section 7.1, page 200.

In the symbolization process, if topological conflicts are created, such clusters with small areas are removed from the data set because the model of generalization considered in this research is the statistical generalization where no conflicts are dealt with as described in Section 8.2, page 255.

Finally, the validation of the two algorithms has been carried out using the ArcGIS software as the benchmark. When comparing the results obtained from the algorithms in the two test areas with that of produced by the ArcGIS software in terms of legibility and preservation constraints based on the qualitative criteria used by the generalization evaluation framework by Stoter *et al.* (2009), the generalized results are more enhanced than that of produced by the proprietary ArcGIS software as discussed in Section 8.2.2. However, the aggregation algorithm developed to deal with non-orthogonal shaped clusters in this research is not suitable to apply on clusters comprising of circular buildings (see Figure 8.23 in Section 8.2.2). The results of ArcGIS software are not satisfactory either in this case.

8.4 Conclusion

The chapter presents the results of focus maps generated in the two test areas - Newham and Tower Hamlets - using the methods and tools implemented in this research in the four fields of spatial data structuring with Delaunay triangulation, data enrichment for subsequent automatic map generalization and salient landmark derivation, data mining for salient landmark derivation and finally automatic map generalization to generate coarse background information in producing the focus maps. The validation of the results of the automatic map generalization and the salient landmarks on the focus maps is carried out using the existing frameworks and the software. A new method based on the MAD is developed to validate landmark saliency in addition to existing methods. Finally, the work done in each field is critically evaluated in relation to existing literature and the results described. The next chapter will present concluding remarks reaffirming results and answers to research questions given in Section 2.8, including remarks on what has been achieved, the overall significance and the areas in the direction for future research.

Chapter 9 Conclusions

This chapter summarises the achievements of the thesis by reaffirming the results and the answers to the research questions. It describes the significance of what has been achieved during the investigation of the processes for generating focus maps for wayfinding. Finally, it presents the areas of future research that have emerged as a result of this research.

9.1 Findings

It is emphasised from the existing work as discussed in Section 2.4.5, page 52 that the CDT algorithm does not provide explicit neighbourhood relationships between polygon objects as it weakens the Delaunay property in applying the edges of polygons as constraints in the data enrichment and the map generalisation applications. There are partial solutions available to this issue according to the existing work: (a) densifying the edges of polygons and apply the CDT and (b) application of the CNDT (see Section 2.4.5, page 53). The first method does not guarantee that the explicit neighbourhood relations are maintained since the densification depends on the distance applied. In the second method, additional points (Steiner points) are added enabling more hooks between polygons to create rich and explicit neighbourhood relations. However, the Steiner points do not exactly lie on the original edges. Thus, if triangles with Steiner points are used to fill the gaps between polygons as discussed in Section 8.1.3, topological exceptions that may be created will have to be resolved. Considering these issues, a new algorithm has been developed and tested in this research to derive explicit neighbourhood relationships between polygon objects without densification or using Steiner points, preserving the Delaunay property thereby retaining original geometries of polygons intact as discussed in Section 4.4. This is achieved initially by applying the Delaunay triangulation on all the vertices (site points) of polygons followed by re-triangulation on the areas created due to the removal of triangle edges that run across polygon edges used as constraints in the triangulation. Since this triangulation algorithm preserves Delaunay property while applying constraints between the edges, it has been termed as DCT in this research. This algorithm is used in the development of two aggregation algorithms in the research as discussed in Sections 7.2.8,

7.3.1 and 8.1.3. However, the DCT is not used in the final clustering algorithm that takes into account linear contextual features such as roads, railways and hydrographic features, as it can only deal with polygon objects. This is despite the DCT having initially been used in clustering buildings during the testing phase of the data enrichment process. Instead, the modified CNDT with the enforcement of edge constraints (see Section 4.2) is used, both on the polygon and the linear features in the clustering process.

In the hierarchical clustering approach developed in the research, the basis has been to derive an adjacency matrix using the Gestalt factors - proximity (the minimum distance), orientation difference and similarity in shape - between each pair of buildings identified using the CNDT with edge constraints. In this process, each node of the triangle edge is enriched with an object IDN that it is connected to derive the relationships between buildings, and between buildings and contextual features by filtering links between them. The algorithm first calculates the Euclidian distance between each known pair of buildings. Then it uses the statistical wall orientation algorithm to derive the orientation difference between each pair of buildings, which has still not been used in the application of building clustering in the literature. Finally, a new similarity index has been developed based on a combination of three measures - overlapping ratio, area to perimeter ratio and discrete Hausdorff distance - between each pair of buildings, depending on the target scale. These pairs of buildings are initially organised into three columns in the adjacency matrix based on the use of the MST segmentation using the proximity as a weight on three ranges of threshold distances (very close, medium and very far) depending on the target scale. Further, the hierarchical splitting is applied to the pairs of buildings in the medium range distance with the MST segmentation with the orientation difference and the similarity index used as weights in order to derive two separate adjacency matrices for holding segmented pairs in terms of the orientation and the similarity differences. Finally, the clustering algorithm developed in this research derives clusters with intelligent cluster classification based on the three values of the Gestalt factors - proximity, orientation difference, and similarity index - between each adjacent pair of buildings as described in Section 5.2.2, page 120. This algorithm has been tested and validated with an experiment of visual cluster perception of buildings with the subjects as discussed in Section 5.3.3. It

can be concluded that the automatic clustering provides improved results in the application of hierarchical clustering when comparing the automatic results generated with this method and the results of the subjects. The results of both expert and lay groups of subjects have not been very satisfactory. This reveals that the hierarchical clustering approach by the subjects involves the retention of considerable information in the mind (high cognitive load) to apply the Gestalt values in an order of sequence in deriving clusters.

When considering the aggregation of building polygons in map generalisation, it is important to preserve the orthogonal shape of the original buildings as well as the shape of the building cluster that is subject to aggregation to provide a meaningful visualisation at the target scale. The other important factor is the way of bridging gaps between building objects when they are placed at exceptional locations such as corner touching, almost overhanging and/or total overhanging positions. Finally, the bridged gaps should be exaggerated so that the amalgam is visually enhanced at the target scale. The two aggregation algorithms developed in this research have provided solutions to all these important factors during aggregation. The aggregation algorithm with orthogonal sides can aggregate buildings in orthogonal clusters (clusters that have buildings with similar orientation and orthogonal sides, touching the cluster outline), preserving orientation and orthogonality in the aggregated building. Further, the aggregation algorithm developed to deal with non-orthogonal shaped clusters (clusters of buildings with non-orthogonal sides and/or dissimilar orientations in the cluster outline) can aggregate buildings, preserving the orthogonality of the edges of buildings (if available) that touch the cluster outline. The most effective capability of both algorithms is that they can aggregate buildings at any exceptional position, exaggerating the narrow corridors and the juts created in bridges during aggregation using a distance threshold in order to provide an enhanced visualisation at the target scale.

As discussed in Section 2.5.3, page 57, the COBWEB unsupervised clustering algorithm and the ID3 supervised classification algorithms have already been used for deriving landmark saliency in existing work. However, they have not been thoroughly tested with real data

sets for this purpose. This research has tested three data mining algorithms - COBWEB, ID3 and J48 (J48 is the implementation of the C4.5 algorithm in WEKA software) - by developing a prototype, modifying the source code of these algorithms given in the WEKA software to traverse through the classification trees in order to derive landmark saliency of building features. It has been identified that C4.5 is the most suitable algorithm to derive landmark saliency when testing with both synthetic and real data sets. Further, the C4.5 algorithm is an improved decision tree over a series of improvements to ID3 as discussed in Section 2.5.1, page 55. Thus, the J48 implementation of C4.5 has been used with real data sets to derive salient landmarks of building features in this research. However, when validating the results as discussed in Section 8.2.3, it has been found that the generated results are not so satisfactory when applied to regions of the test data sets and at a decision point. Therefore, the overall implication is that this decision tree algorithm is to be further improved and modified for deriving landmark saliency.

The results of focus maps are validated in two phases. In the first phase, the evaluation of the results of the generalized background of building features on the focus maps produced in the two test areas is compared to that generated using the ArcGIS proprietary software in terms of the preservation and the legibility constraints based on the existing framework by Stoter *et al.* (2009) as discussed in Section 3.3.3, page 73. When observing the evaluated results as discussed in Section 8.2.2, it can be concluded that the results of generalisation produced in this research have been very promising and much more satisfactory than that produced by the ArcGIS software.

In the second phase, the derived landmark saliency with the J48 implementation is evaluated with three frameworks - two existing frameworks by Raubal and Winter (2002) and Nothegger, Winter and Raubal (2004), and the new framework developed in this research. In the first framework by Raubal and Winter (2002), the data are assumed to be normally distributed while the other two frameworks consider non-normality of data. As a cross-check of the results, the Google street views of the salient landmarks in the two test areas are used to compare the results given by each framework. In the application of the framework by Raubal and Winter (2002), it is found that the results are impacted by the

outliers since the data is assumed to be normally distributed. Further, when validating with the framework by Nothegger, Winter and Raubal (2004), the saliency score (which is equivalent to a Z-score) based on the MAD of data provides incorrect results - scoring of infinity - when applied to the test data, although this framework is more resistant in outlier detection as discussed in Section 8.2.3. Further, the values derived with this score tend to be higher for the overall significance measure. This measure can be used to identify the most salient features at a decision point. In the case of detecting landmarks in a region, it is difficult to choose a particular critical threshold with these higher values of the significance measure to detect landmark saliency. However, the new method developed on the MAD proves to give promising results for deriving landmark saliency over all the other methods, including the J48 implementation both in the regions and at decision points as discussed in Section 8.2.3. However, the decision criterion of the MAD used in this research checks only outliers in the upper limit in order to avoid selecting buildings as outliers over attributes sensitive to smaller values such as the small size of buildings. As a result, the important buildings due to low heights would not be chosen as outliers to represent salient buildings. Further testing is necessary to improve this framework with the decision criterion on the MAD to detect outliers.

9.2 Contribution

- How is the CDT data structure used and validated to derive explicit neighbourhood relationships between building polygons?

The triangulation algorithm developed can be used to retrieve explicit adjacency relationships between polygons with complex geometries such as properties of shared edges and holes and their proximity relations for applications involving data enrichment and automatic map generalisation.

- How building objects are clustered with the hierarchical application of Gestalt factors, considering contextual features with an intelligent cluster classification for the subsequent generalisation process?

The hierarchical polygon clustering algorithm developed to support subsequent automatic map generalisation enables the choice of application of different generalisation operators based on the characteristics of each cluster. It has the ability to constrain the clustering process using contextual features such as roads, railways and hydrographic features. Further, since the clustering approach is hierarchical, different clustering results can be generated based on the representation of the same phenomena by applying different threshold values to support generalisation at different resolution levels. This clustering algorithm can also be used for the automatic map generalisation in other applications involving clustering of polygon objects based on their characteristics assigned to each polygon. The characteristics used in this research are only the Gestalt factors - proximity, orientation difference and the similarity difference in shape - between polygons.

- Which generalisation algorithms are the most influential for the merging (aggregation) of building clusters depending on their outline shape (orthogonal or irregular) in order to provide a meaning to merging at coarser representation levels?

The two building aggregation algorithms developed in this research can be used to aggregate building clusters, taking into consideration of the cluster characteristics to generate building amalgams, preserving the characteristics of shape and orientation of the source clusters in any generalisation application involving building aggregation. These two algorithms have been developed to handle almost every exception of building positions in a cluster such as corner touching and overhanging positions of buildings. Further, both algorithms have the facility to exaggerate the gaps filled during aggregation, taking into account the target scale of the amalgams in order to enhance their legibility at the target generalized scale.

- Which data mining algorithm is the most appropriate to emphasise building landmark saliency?

The J48 implementation of the C4.5 decision tree algorithm used to derive landmark saliency in the regions and at decision points is not satisfactory when compared to the method based on the MAD developed to validate the results of the J48 implementation.

Finding the statistical significance based on the MAD is a promising method to derive both on-route landmarks and landmarks at decision points over the data mining algorithms according to the analysis of the results of this research. The research has also developed algorithms and methods to extract, especially the visual and the structural characteristics of building features, taking into consideration the context for the subsequent derivation of salient buildings to be incorporated as landmarks in wayfinding focus maps.

- How are the focus maps validated in terms of the generalized results and the derived landmark saliency?

There are two types of validation involved in this research - one is to validate the results of the generalized output used as coarse background on focus maps and the other is to validate the saliency of the derived landmarks. Only a qualitative evaluation is carried out based on the visual comparison of the generalized results produced using the developed algorithms and the tools available in the proprietary ArcGIS software for similar generalization. This evaluation is one of the three methods given in the generalisation evaluation framework of Stoter *et al.* (2009). The two other methods - a qualitative evaluation by cartographic experts and an automated constrained-based evaluation - are promising methods. However, considering the time and the cost involved, the research does not use these two methods. A thorough validation is carried out by implementing the two existing frameworks by Raubal and Winter (2002) and Nothegger, Winter and Raubal (2004) for the evaluation of landmark saliency derived from the J48 implementation. A new method based on the MAD is also developed and implemented for the landmark validation. In addition, a further cross-check of the landmark saliency results using the J48 implementation, the framework of Raubal and Winter (2002) and the method based on the MAD is carried out with the Google street view.

Finally, with the combination of all the algorithms and the methods developed at each stage of the four fields: triangulation data structure, data enrichment, automatic map generalisation and knowledge discovery of salient landmarks, this research has been able to fill in the gaps in generating focus maps through the investigation of the processes involved.

9.3 Future research

The DCT algorithm developed and used here can only deal with retrieving adjacency relationships between polygon features. This structure should further be extended to deal with both polygon and linear features.

The aggregation algorithm developed to merge buildings in orthogonal shaped clusters presently cannot deal with squaring and simplification of holes that need to be retained in the target scale, depending on their size (area) using a threshold. Therefore, the present algorithms should be extended to incorporate and generalize holes if they are significant to be represented on the target scale. The aggregation algorithm developed to deal with buildings belonging to non-orthogonal clusters currently may fill concave corners in the building outline with self-connecting bridges when applying the buffering technique with a distance equal to the building clustering threshold (see Section 7.3.1, page 230). Further work should be carried out as to how such corners are to be preserved without filling such corners during the aggregation process.

Although the decision tree algorithm (C4.5) used in this research for deriving landmark saliency is robust in terms of dealing with all the types of attributes, further work is necessary to modify the tree structure so that it can identify the most discerning salient features, taking into consideration visual, structural and semantic measures of features. The framework developed to validate saliency results given by the J48 implementation of the C4.5 algorithm using the application of the MAD suggested by Leys *et al.* (2013) should further be tested on building features with different threshold values for the Z-score (value 2.5 has been used in this research) for outlier detection using both lower and upper limits of the decision criterion by considering the attribute transformation to enable the effective detection of outliers for deriving salient landmarks in wayfinding applications.

Bibliography

- Adam, B., Kauffmann, P., Schmitt, D. and Spehner, J. (1997) 'An increasing circle sweep algorithm to construct the Delaunay diagram in the plane', *In: Proceedings of Canadian Conference on Computational Geometry*.
- Ai, T., Zhang, X., Fabrikant, S. I. and Wachowicz, M. (2007) 'The Aggregation of urban building clusters based on the skeleton partitioning of gap space', The European Information Society. Springer Berlin Heidelberg, pp. 153-170.
- Allen, G. (1999) 'Spatial abilities, cognitive maps, and wayfinding - bases for individual differences in spatial cognition and behaviour'. in Golledge, R. (ed.) *Wayfinding Behavior - Cognitive Mapping and Other Spatial Processes*. Johns Hopkins University Press, Baltimore, pp. 46-80.
- Allouche, M. K. and Moulin, B. (2005) 'Amalgamation in cartographic generalization using Kohonen's feature nets', *International Journal of Geographical Information Science*, 19(8/9), pp. 899-914. doi: 10.1080/13658810500161211.
- Anders, K.-H., Riedl, A., Kainz, W. and Elmes, G. A. (2006) 'Grid typification'. *Progress in Spatial Data Handling*. Springer Berlin Heidelberg, pp. 633-642.
- Anders, K. H. (2003) 'A hierarchical graph-clustering approach to find groups of objects', *In: Proceedings of 5th Workshop on progress in Automated map generalization, Paris, France, 2003*.
- ArcGIS (2013) 'esri'. Available at: <http://www.esri.com/software/arcgis> (Accessed 21st May 2013).
- Armstrong, M. P. (1991) 'Knowledge classification and organization'. in Buttenfield, B.P. and MacMaster, R.B. (eds.) *Map Generalization - Making rules for knowledge representation*. Essex, England: Longman group, pp. 86-102.
- Atallah, M. J. (1983) 'A linear time algorithm for the Hausdorff distance between convex polygons', *Information Processing Letters*, 17, pp. 207-209.
- Bader, M., Barrault, M., Regnault, N., Mustière, S., Duchêne, C., Ruas, A., Barillot, X. (1999) 'AGENT work package D2 - selection of basic algorithms', Technical Report, Department of Geography, University of Zurich.
- Bader, M. and Weibel, R. (1997) 'Detecting and resolving size and proximity conflicts in the generalization of polygonal maps', *In: Proceedings of 18th International Cartographic Conference (ICC), Stockholm, pp. 1525-1532*.
- Baker, B., Grosse, E. and Rafferty, C. (1988) 'Nonobtuse triangulation of polygons', *Discrete & Computational Geometry*, 3(1), pp. 147-168. doi: 10.1007/BF02187904.
- Bard, S. (2003) 'Evaluation of generalization quality', *6th ICA workshop on generalization and multiple representation, 28-30 April, Paris, France*.
- Bard, S. (2004) 'Quality assessment of cartographic generalisation', *Transactions in GIS*, 8(1), pp. 63-81. doi: 10.1111/j.1467-9671.2004.00168.x.

- Barrault, M., Regnauld, N., Duchene, C., Haire, K., Baeijs, C., Demazeau, Y., Weibel, R. (2001) 'Integrating multi-agent, object-oriented and algorithmic techniques for improved automated map generalization', In: *Proceedings of 20th International Cartographic Conference (ICC), Beijing*, pp. 2110-2116.
- Basaraner, M. and Selcuk, M. (2004) 'An attempt to automated generalization of buildings and settlement areas in topographic maps', In: *Proceedings of 20th ISPRS congress, Istanbul, Turkey 2004*.
- Basaraner, M. and Selcuk, M. (2008) 'A structure recognition technique in contextual generalisation of buildings and built-up areas', *Cartographic Journal*, 45(4), pp. 274-285. doi: 10.1179/174327708x347773.
- Beard, M. (1991) 'Constraints on rule formation'. in Battenfield, B. and McMaster, R. (eds.) *Map generalization: making rules for knowledge representation*. Longman, London, pp. 121-135.
- Beard, M. K. (1987) 'How to survive on a single detailed database', In: *Proceedings of Auto-Carto 8*. Baltimore, Maryland. pp. 211-220.
- Bereuter, P. and Weibel, R. (2010) 'Generalization of point data for mobile devices: A problem oriented Approach'. *International Cartographic Association (ICA) workshop on Generalization and Multiple Representation, 13th September 2010, Zurich*.
- Berg, M., de, Cheong, O., Kreveld, M., van and Overmars, M. (2008) 'Delaunay triangulations', *Computational Geometry: Algorithms and Applications*. 3rd Edition (March 2008) edn.: Springer-Verlag, 99. pp. 191-216.
- Bern, M., Eppstein, D. and Gilbert, J. (1994) 'Provably good mesh generation', *Journal of Computer and System Sciences*, 48(3), pp 384-409. doi: [http://dx.doi.org/10.1016/S0022-0000\(05\)80059-5](http://dx.doi.org/10.1016/S0022-0000(05)80059-5).
- Biniarz, A. and Dastghaibifard, G. (2012) 'A faster circle-sweep Delaunay triangulation algorithm', *Advances in Engineering Software*, 43(1), pp. 1-13.
- Brassel, K. E. and Weibel, R. (1988) 'A review and conceptual framework for automated map generalization', *International Journal of Geographic Information Systems*, 2(3), pp. 229-244.
- Brenner, C. and Elias, B. (2003) 'Extracting landmarks for car navigation systems using existing GIS databases and laser scanning', *PIA 2003, ISPRS Archives, Vol. XXXIV, Part 3/W8, Munich, 17 -19 Sept. 2003*.
- Brimicombe, A. and Li, C. (2010) '*Location-based services and geo-information engineering (Mastering GIS: Technology, Applications and Management)*', Chichester: Wiley.
- Brimicombe, A. J. and Li, Y. (2006) 'Mobile space-time envelopes for location-based services', *Transactions in GIS*, 10(1), pp. 5-23. doi: 10.1111/j.1467-9671.2006.00241.x.
- Burnett, G., Smith, D. and May, A. (2001) 'Supporting the navigation task: characteristics of good landmarks', in Hanson, M.A. (ed.) *Contemporary Ergonomics 2001*. London: Taylor and Francis, pp. 441-446.

Buttenfield, B. P. and McMaster, R. B. (1991) *Map generalization: making rules for knowledge representation*. Essex, England: Longman group, pages 245.

Cecconi, A. (2003) *Integration of cartographic generalization and multi-scale databases for enhancement web mapping, Doctoral dissertation*. University of Zurich, pages 138.

Chaudhry, O. and Mackaness, W. (2006) 'Indeterminate boundaries in higher order phenomenon', *International Cartographic Association (ICA) workshop on generalization and multiple representation, Portland, USA, 25th June 2006*.

Chazelle, B. (1991) 'Triangulating a simple polygon in linear time', *Discrete and Computational Geometry*, 6, pp. 485-524.

Chazelle, B. and Chazelle, B. (1982) 'A theorem on polygon cutting with applications', *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium*, pp. 339-349. doi: 10.1109/SFCS.1982.58.

Chew, L. P. (1989) *Guaranteed-quality triangulation meshes, Technical report, No. TR-89-983*. Cornell university.

Christophe, S. and Ruas, A. (2002) 'Detecting building alignments for generalization purposes', in Richardson, D.E. and Oosterom, P.v. (eds.) *Advances in spatial data handling (The 10th International symposium on spatial data handling)*. Berlin, Heidelberg New York: Springer, pp. 459-475.

Concave hull (2013) 'Concave hull based on JTS'. Available at: http://www.rotefabrik.free.fr/concave_hull (Accessed: 11th April 2013).

Cormen, T., H., Leiserson, C., E., Rivset, R., L. and Stein, C. (2009) 'Introduction to algorithms'. Third edn. Massachusetts Institute of Technology, pp. 253-280.

Cornell, E. H., Sorenson, A. and Mio, T. (2003) 'Human sense of direction and wayfinding', *Annals of the Association of American Geographers*, 93(2), pp. 399-425. doi: 10.1111/1467-8306.9302009.

Deakin, A. K. (1996) 'Landmarks as navigational aids on street maps', *Cartography and Geographic Information Science*, 23(1), pp. 21-36.

Delaunay, B. (1934) 'Delaunay triangulation', Wikipedia. Available at: http://en.wikipedia.org/wiki/Delaunay_triangulation (Accessed 9th July 2011).

DeLucia, A. and Black, T. (1987) 'A comprehensive approach to automatic feature generalization', In: *Proceedings of the International Cartographic Conference (ICC)*, Morelia, Mexico, pp. 169-192.

Denis, M., Pazzaglia, F., Cornoldi, C. and Bertolo, L. (1999) 'Spatial discourse and navigation: an analysis of route directions in the city of Venice', *Applied Cognitive Psychology*, 13(2), pp. 145-174.

Domiter, V. and Žalik, B. (2008) 'Sweep-line algorithm for constrained Delaunay triangulation', *International Journal of Geographical Information Science*, 22(4), pp. 449-462. doi: 10.1080/13658810701492241.

- Douglas, D. H. and Peucker, T. K. (1973) 'Algorithms for the reduction of the number of points required to represent a digitized line or its character', *The Canadian cartographer*, 10(2), pp. 112-123.
- Duchêne, C., Bard, S., Barillot, X., Ruas, A., Trévisan, J. and Holzapfel, F. (2003) 'Quantitative and qualitative description of building orientation'. *The 5th International Cartographic Association (ICA) workshop in automated map generalization*. Paris, France.
- Duckham, M., Kulik, L., Worboys, M. and Galton, A. (2008) 'Efficient generation of simple polygons for characterizing the shape of a set of points in the plane', *Pattern Recognition*, 41(10), pp. 3224-3236. doi: 10.1016/j.patcog.2008.03.023.
- Dunkars, M. (2004) *Multiple representation databases for topographic information*. Department of infrastructure, Royal Institute of technology (KTH), Doctoral dissertation in Geo-informatics, pages 183.
- Dwyer, R., A. (1987) 'A faster-divide-and-conquer algorithm for constructing Delaunay triangulations', *Algorithmica*, 2, pp. 137-151.
- Eberly, D. (2008) 'Triangulation by ear clipping.' pages 13. Available at: <http://www.geometrickit.com/> (Accessed 11th August 2012).
- Egenhofer, M., Litwin, W. and Schek, H.-J. (1989) 'A formal definition of binary topological relationships', *Foundations of Data Organization and Algorithms*. Springer Berlin / Heidelberg, pp. 457-472.
- ElGindy, H., Everett, H. and Toussaint, G. (1993) 'Slicing an ear using prune-and-search', *Pattern Recognition Letters*, 14(9), pp. 719-722.
- Elias, B. (2003) 'Extracting landmarks with data mining methods', In *Spatial Information Theory: Foundations of Geographic Information Science, Vol. 2825, Lecture Notes in Computer Science*, W. Kuhn, M.F. Worboys, and S. Timpf, Eds. Berlin: Springer, pp. 398-412.
- Elias, B. and Brenner, C. (2005) 'Automatic generation and application of landmarks in navigation data sets'. *Developments in Spatial Data Handling*. Springer Berlin Heidelberg, pp. 469-480.
- Elias, B., Hampe, M. and Sester, M. (2005) 'Adaptive visualization of landmarks using an MRDB'. *Map-based Mobile Services - Theories, Methods and Implementations*. Springer Berlin Heidelberg, pp. 73-86.
- Elias, B. and Paelke, V. (2008) 'User-centered design of landmark visualizations'. *Map-based Mobile Services - Design, Interaction and Usability*.: Springer Berlin Heidelberg, pp. 33-56.
- Elias, B. and Sester, M. (2006) 'Incorporating landmarks with quality measures in routing procedures'. in Raubal, M., Miller, H.J., Frank, A.U. and Goodchild, M.F. (eds.) *Geographic Information Science*. Springer Berlin Heidelberg, pp. 65-80.
- Erten, H. and Üngör, A. (2009) 'Quality triangulations with locally optimal Steiner points', *SIAM Journal on Scientific Computing*, 31(3), pp. 2103-2130. doi: 10.1137/080716748.

- Ester, M., Kriegel, H.-P. and Sander, J. (1997) 'Spatial data mining: A database approach', In: *Proceedings of 5th International Symposium on large spatial databases, Berlin, Germany, Lecture notes in Computer Science, Springer, 1997.*
- Fairbairn, D. and Taylor, G. (1995) 'Developing a variable-scale map projection for urban areas', *Computers & Geosciences*, 21(9), pp. 1053-1064.
- Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) From Data Mining to Knowledge Discovery in Databases, AAAI Press. *Artificial Intelligence*, pp. 37-54.
- Fisher, D. H. (1987) 'Knowledge Acquisition via Incremental conceptual clustering', *Machine Learning*, 2(2), pp. 139-172. doi: 10.1023/A:1022852608280.
- Fortune, S. (1987) 'A sweep line algorithm for Voronoi diagrams', *Algorithmica*, 2(1-4), pp. 153-174. doi: 10.1007/BF01840357.
- Frawley, W. J., Piatetsky-Shapiro, G. and Matheus, C. J. (1992) Knowledge discovery in databases: An overview. *AI*, 13, (3), pp. 57-70.
- Freksa, C., Mark, D., Fontaine, S. and Denis, M. (1999) 'The production of route instructions in underground and urban environments'. *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*. Springer Berlin Heidelberg, pp. 83-94.
- Freksa, C., Mark, D., Tversky, B. and Lee, P. (1999) 'Pictorial and verbal tools for conveying routes'. *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*. Springer Berlin Heidelberg, pp. 51-64.
- Julian, G., (2009) 'Three reuse examples of a generic deformation model in map generalization', *24th International Cartographic Conference, 15-21 November, Santiago, Chile.*
- Galanda, M. (2003) 'Automated polygon generalization in a multi agent system', Publication of the university of Zurich, Doctoral dissertation, Pages 176.
- Gartner, G. and Uhlirz, S. (2001) 'Cartographic concepts for realizing a location based UTMS service: Vienna City Guide "LoL@"', In: *Proceedings of 20th International Cartographic Conference (ICC), 6-10, August, Beijing, China.*
- Gennari, J. H., Langley, P. and Fisher, D. (1989) 'Models of incremental concept formation', *Artificial Intelligence*, 40, pp. 11-61.
- Golledge, R. (1999) 'Human wayfinding and cognitive maps', *Wayfinding Behavior: Cognitive Mapping and other Spatial Processes*. John Hopkins Press, pp. 5-46.
- Gopal, S. and Smith, T. R. (1990) 'Human way-finding in an urban environment: a performance analysis of a computational process model', *Environment and Planning A*, 22(2), pp. 169-191.
- Guibas, L., Knuth, D. and Sharir, M. (1992) 'Randomized incremental construction of Delaunay and Voronoi diagrams', *Algorithmica*, 7(1), pp. 381-413. doi: 10.1007/BF01758770.

- Günter, R. (1991) 'Computing the minimum Hausdorff distance between two point sets on a line under translation', *Information Processing Letters*, 38(3), pp. 123-127.
- Guttman, A. (1984) 'R-Trees: A Dynamic Index structure for spatial searching'. in Yormark, B. (ed.) *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984* ACM Press, pp. 47-57.
- Hampe, M., Anders, K.-H. and Sester, M. (2003) 'MRDB applications for data revision and real-time generalization'. *21st International Cartographic Conference, 10-16, August 2003, Durban, South Africa, 2003*. 10-16. August 2003, Durban, South Africa, 2003.
- Hampe, M. and Sester, M. (2004) 'Generating and using a multi-representation database (MRDB) for mobile applications'. In: *Proceedings of the 8th ICA workshop on generalisation and multiple representation*. Leicester, 20–21 August 2004.
- Hampe, M., Sester, M. and Harrie, L. (2004) 'Multiple representation databases to support visualization on mobile devices', In: *Proceedings of 20th ISPRS congress, Istanbul, Turkey 2004*.
- Haowen, Y., Weibel, R. and Bisheng, Y. (2008) 'A Multi-parameter approach to automated building grouping and generalization', *Geoinformatica*, 12(1), pp. 73-89. doi: 10.1007/s10707-007-0020-5.
- Hirtle, S. C. (1985) 'Evidence of hierarchies in cognitive maps', *Memory & cognition*, 13(3), pp. 208-217.
- Hjelle, Ø. and Dæhlen, M. (2006) *Triangulations and applications*. Berlin, Heidelberg: Springer.
- Huber, P. J. (1981) *Robust Statistics*, Wiley, New York NY.
- Jaromczyk, J. W. and Toussaint, G. T. (1992) 'Relative neighborhood graphs and their relatives', In: *Proceedings of the IEEE*, 80(9), pp. 1502-1517. doi: 10.1109/5.163414.
- Jiang, B. and Yao, X. (2006) 'Location-based services and GIS in perspective', *Computers, Environment and Urban Systems*, 30(6), pp. 712-725.
- Jones, C., Bundy, G., Li and Ware, J., Mark (1995) 'Map generalization with a triangulated data structure', *Cartography and Geographic Information Systems*, 22(4), pp. 317-331.
- Jones, C., B. and Mark, W., J. (1998) 'Proximity search within a triangulated spatial model', *The Computer journal*, 41(2).
- João, M. E. (1998) *Causes and consequences of map generalization*. London: Taylor and Francis, pages 266.
- JTS Topology Suite (2012) 'JTS user forum'. Available at: <http://sourceforge.net/p/jts-topo-suite/mailman/jts-topo-suite-user/> (Accessed: 3rd January 2012).
- Kilpelinen, T. (1997) *Multiple representation and generalization of geo-databases for topographic maps*. Finnish Geodetic Institute, Doctoral dissertation, pages 229.

- Kolingerová, I. and Žalik, B. (2002) 'Improvements to randomized incremental Delaunay insertion', *Computers & Graphics*, 26(3), pp. 477-490. doi: [http://dx.doi.org/10.1016/S0097-8493\(02\)00090-0](http://dx.doi.org/10.1016/S0097-8493(02)00090-0).
- Krakk, M. and Ormeling, F. (2003) '*Cartography: visualization of geospatial data*', Second edition. Edinburgh Gate, Harlow, Essex CM 20 2JE, England: Pearson Education Limited, pp. 75-84.
- Kray, C., Elting, C., Laakso, K. and Coors, V. (2003) 'Presenting route instructions on mobile devices'. in: *Proceedings of the 8th International Conference on Intelligent user interfaces*. Miami, Florida, USA. ACM.DOI: 10.1145/604045.604066.
- Kuipers, B. (1978) 'Modeling spatial knowledge', *Cognitive Science*, 2(2), pp. 129-153.
- Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W. A. and Weibel, R. (1999) 'The application of agents in automated map generalization', In: *Proceedings of the 19th International Cartographic Conference (ICC), Ottawa, Canada*, pp.1225-1234.
- Leutenegger, S. T., Lopez, M. A. and Edgington, J. (1997) 'STR: a simple and efficient algorithm for R-tree packing', In: *Proceedings. 13th International Conference on Data Engineering*, 1997, pp. 497-506.
- Leys, C., Ley, C., Klein, O., Bernard, P. and Licata, L. (2013) 'Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median', *Journal of Experimental Social Psychology*, 49(4), pp. 764-766. doi: <http://dx.doi.org/10.1016/j.jesp.2013.03.013>.
- Li, C. (2006) 'User preferences, information transactions and location-based services: A study of urban pedestrian wayfinding', *Computers, Environment and Urban Systems*, 30(6), pp. 726-740.
- Li, Z. (1994) 'Mathematical morphology in digital generalization of raster map data', *Cartographica*, 23, pp. 1-10.
- Li, Z., Yan, H., Ai, T. and Chen, J. (2004) 'Automated building generalization based on urban morphology and Gestalt theory', *International Journal of Geographical Information Science*, 18(5), pp. 513-534. doi: 10.1080/13658810410001702021.
- Lichtner, W. (1979) 'Computer-assisted processes of cartographic generalization in topographic maps', *Geo-Processing*, 1, pp. 183-199.
- Lipton, R. J. and Tarjan, R. E. (1979) 'A separator theorem for planar graphs', *SIAM J. Comput.*, 36, pp. 177-189.
- Liu, Y., Molenaar, M., Ai, T. and Liu, Y. (2003) 'Categorical database generalization aided by data model', In: *Proceedings of the 21st International Cartographic Conference (ICC), Durban, South Africa, 10-16 August 2003*.
- Lloyd, R. (1989) 'Cognitive maps: encoding and decoding information', *Annals of the Association of American Geographers*, 79(1), pp. 101-124.
- Lovelace, K., Hegarty, M. and Montello, D. (1999) 'Elements of good route directions in familiar and unfamiliar environments'. in Freska, C. and Mark, D. (eds.) *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*. Springer Verlag, pp. 65-82.

Lynch, K. (1960) '*The Image of the City*'. MIT Press, Cambridge.

MacEachren, A. (1995) '*How maps work*'. The Guildford Press.

Mackaness, W., A. and Beard, M., K. (1993) 'Use of graph theory to support generalization', *Cartography and Geographic Information Systems*, 20, pp. 210-221.

Mackaness, W. A. (1995) 'A constraint based approach to human computer interaction in automated cartography', In: *Proceedings of International Cartographic Conference (ICC), Barcelona, Spain*, pp. 1423-1432.

May, A. J., Ross, T., Bayer, S. H. and Tarkiainen, M. J. (2003) 'Pedestrian navigation aids: information requirements and design implications', *Personal Ubiquitous Comput.*, 7(6), pp. 331-338. doi: 10.1007/s00779-003-0248-5.

McMaster, R. B. (1987) 'Automated line generalization', *Cartographica: The International Journal for Geographic Information and Geovisualization*, 24(2), pp. 74-111. doi: 10.3138/3535-7609-781G-4L20.

McMaster, R. B. and Shea, K. S. (1992) *Generalization in digital cartography*. Association of American Geographers. Resource publications in Geography. Washington, D.C.

Meisters, G. H. (1975) 'Polygons have ears', *Mathematical Monthly*, 82(6), pp. 648-651.

Melissaratos, E. A. and Souvaine, D. L. (1992) 'Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes', in: *Proceedings of the eighth annual Symposium on Computational Geometry*. Berlin, Germany. ACM.DOI: 10.1145/142675.142719.

Meng, L. (2005) 'Ego centres of mobile users and egocentric map design'. *Map-based Mobile Services*. Springer Berlin Heidelberg, pp. 87-105.

Meng, L. and Reichenbacher, T. (2005) 'Map-based mobile services', *Map-based Mobile Services*. Springer Berlin Heidelberg, pp. 1-8.

Mennis, J. and Guo, D. (2009) 'Spatial data mining and geographic knowledge discovery—An introduction', *Computers, Environment and Urban Systems*, 33(6), pp. 403-408. doi: <http://dx.doi.org/10.1016/j.compenvurbsys.2009.11.001>.

Miller, H. J. and Han, J. (2009) *Geographic data mining and Knowledge discovery*. 2nd Edition. CRC press, Boca Raton FL.

Miller, J. (1991) 'Short report: Reaction time analysis with outlier exclusion: Bias varies with sample size', *The Quarterly Journal of Experimental Psychology Section A*, 43(4), pp. 907-912. doi: 10.1080/14640749108400962.

Montello, D., Michon, P.-E. and Denis, M. (2001) 'When and why are visual landmarks used in giving directions?'. *Spatial Information Theory*. Springer Berlin Heidelberg, pp. 292-305.

- Morgenstern, D. and Schürer, D. (1999) 'A concept for model generalization of digital landscape models from finer to coarser resolution', In: *Proceedings of International Cartographic Conference (ICC)*. Ottawa, Canada.
- Nam, N. M., Kiem, H. V. and Nam, N. V. (2009) 'A fast algorithm for constructing constrained Delaunay triangulation', *Computing and Communication Technologies, 2009. RIVF '09. International Conference on*, pp. 1-4.
- Neis, P. and Zipf, A. (2008) 'Extending the OGC OpenLS route service to 3D for an interoperable realisation of 3D focus maps with landmarks', *Journal of Location Based Services*, 2(2), pp. 153-174. doi: 10.1080/17489720802400413.
- Neun, M., Weibel, R. and Burghardt, D. (2004) 'Data enrichment for adaptive generalisation'. *International Cartographic Association (ICA) Workshop on Generalisation and Multiple representation, 20-21 August 2004, Leicester*.
- Nivala, A., Sarjakoski, L. T., Jakobsson, A. and Kaasinen, C. (2003) 'Usability evaluation of topographic maps in mobile devices', In: *Proceedings of 21st International Cartographic Conference (ICC)*, Durban, South Africa, 10 - 16, August 2003, pp. 1903-1914.
- Nothegger, C., Winter, S. and Raubal, M. (2004) 'Computation of the salience of features', *Spatial Cognition and Computation*, 4(2), pp. 113-136.
- Oosterom, P. v. (2009) 'Research and development in geo-information generalisation and multiple representation', *Computers, Environment and Urban Systems*, 33(5), pp. 303-310. doi: 10.1016/j.compenvurbsys.2009.07.001.
- OpenCarto (2013) 'OpenCarto Java library'. Available at: <http://opencarto.net23.net> (Accessed 02nd January 2013).
- Ormsby, D. and Mackaness, W. (1999) 'The development of phenomenological generalization within an object-oriented paradigm', *Cartography and Geographic Information Science*, 26(1), pp. 70-80.
- Palmer, S. and Rock, I. (1994) 'Rethinking perceptual organization: The role of uniform connectedness', *Psychonomic Bulletin & Review*, 1(1), pp. 29-55. doi: 10.3758/BF03200760.
- Palmer, S. E. (1992) 'Common region: a new principle of perceptual grouping', *Cognitive Psychology*, 24, pp. 436-447.
- Peng, W. (1997) *Automated generalization in GIS, Doctoral dissertation*. International Institute for Aerospace Survey and Earth Sciences (ITC), the Netherlands.
- Poly2tri (2012) 'A 2D constrained Delaunay triangulation library'. Available at: <http://code.google.com/p/poly2tri> (Accessed: 7th March 2012).
- PostGIS (2013) 'Spatial and geographic objects for PostgreSQL'. Available at: <http://postgis.net/> (Accessed 14th March 2013).
- PostGIS Manual (2012) 'PostGIS 2.2.0dev Manual'. Available at: <http://postgis.net/docs/manual-dev/> (Accessed: 10th August 2012).

Presson, C. C. and Montello, D. R. (1988) 'Points of reference in spatial cognition: Stalking the elusive landmark*', *British Journal of Developmental Psychology*, 6(4), pp. 378-381. doi: 10.1111/j.2044-835X.1988.tb01113.x.

Prim, R. (1957) 'Shortest connection networks and some generalizations', *Bell System Technical Journal*, 36, pp. 1389-1401.

Pyle, D. (1999) *Data preparation for data mining*. ed. Serra, D.D., Morgan Kaufmann Publishers, Inc., Editorial and Sales Office, 340, Pine Street, Sixth floor, San Francisco, CA 94104-3205, USA.

Qi, H., B. and Li, Z., L. (2008) 'An approach to building grouping based on hierarchical constraints', *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII, Part B2.

Quinlan, J. R. (1986) 'Induction of decision trees', *Machine Learning*, 1(1), pp. 81-106. doi: 10.1023/a:1022643204877.

Quinlan, J. R. (1993) *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc.

Radoczky, V. and Gartner, G. (2005) 'Extent and effectiveness of map abstraction for communicating routes in a LBS', in: *Proceedings of 22nd International Cartographic Conference, 9-16 July 2005, La Coruña, Spain*.

Rainsford, D. and Mackaness, W. (2002) 'Template matching in support of generalisation of rural buildings'. in Richardson, D. and Oosterom, P. (eds.) *Advances in Spatial Data Handling*. Springer Berlin Heidelberg, pp. 137-151.

Raubal, M. and Winter, S. (2002) 'Enriching wayfinding instructions with local landmarks'. in: *Proceedings of the Second International Conference on Geographic Information Science*, Springer-Verlag, pp. 243-259.

Regnauld, N. (1996) 'Recognition of building clusters for generalization'. in Kraak, M.J. and Molenaar, M. (eds.) *Advances in GIS II (Proceedings of the 7th International Symposium on spatial data handling, Session 4B)*. London: Taylor and Francis, pp. 185-198.

Regnauld, N. (2001) 'Contextual building typification in automated map generalization', *Algorithmica*, 30(2), pp. 312-333. doi: 10.1007/s00453-001-0008-8.

Regnauld, N. (2003) 'Algorithms for the amalgamation of topographic data', In: *Proceedings of the 21st International Cartographic Conference (ICC), Durban*.

Regnauld, N. (2005) 'Spatial structures to support automatic generalization', In: *Proceedings of the 22nd International Cartographic conference (ICC), La Coruna*.

Regnauld, N. and Revell, P. (2007) 'Automatic amalgamation of buildings for producing Ordnance Survey[®] 1:50 000 scale maps', *Cartographic Journal*, 44(3), pp. 239-250. doi: 10.1179/000870407x241782.

Reichenbacher, T. (2003) 'Adaptive methods for mobile cartography', In: *Proceedings of 21st International Cartographic Conference, 10-16 August 2003, Durban, South Africa*.

Reichenbacher, T. (2004) *Mobile Cartography - Adaptive visualization of geographic information on mobile devices*. Doctoral Dissertation, Department of Cartography, Technical University of München.

Reichenbacher, T. (2005) 'Adaptive egocentric maps for mobile users'. *Map-based Mobile Services - Theories, Methods and Implementations*. Springer Berlin Heidelberg, pp. 141-158.

Revell, P. (2008) 'A review of the Clarity generalization platform and the customizations developed at Ordnance Survey research', *11th International Cartographic Association (ICA) workshop on Generalization and Multiple representation*. 20th to 21st June 2008, Montpellier.

Revell, P. and Antoine, B. (2009) 'Automated matching of building features of differing levels of detail: A case study'. *24th International Cartographic Conference, Santiago de Chile*.

Revell, P., Regnaud, R. and Thomas, S. G. (2005) 'Generalizing OS MasterMap topographic buildings and ITN road centrelines to 1:50 000 scale using a spatial hierarchy of agents, triangulation and topology', in: *Proceedings of the 22nd International Cartographic Conference (ICC)*. La Courña, Spain.

Rieger, M. K. and Coulson, M. R. C. (1993) 'Consensus or confusion: Cartographers' knowledge of generalization', *Cartographica: The International Journal for Geographic Information and Geovisualization*, 30(2), pp. 69-80. doi: 10.3138/M6H4-1006-6422-H744.

Robinson, H. A., Morrison, J. L., Muehrke, P. C., Kimerling, A. J. and Guptill, S. C. (1995) *Elements of cartography*, 6 edn. New York: Wiley.

Ruas, A. (1995) 'Multiple paradigms for automated map generalization: Geometry, topology, hierarchical partitioning and local triangulation', in: *Proceedings AutoCarto 12*, pp. 69-78, Charlotte, USA.

Ruas, A. and Plazanet, C. (1996) 'Strategies for automated generalization'. In Kraak, M.J. and Molenaar, M. (eds.) *Advances in GIS research II, (In: Proceedings of the 7th International Symposium on Spatial Data handling)*. Taylor & Francis, London, pp. 6.1-6.18.

Ruppert, J. (1993) 'A new and simple algorithm for quality 2-dimensional mesh generation', in: *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, Austin, Texas, USA, 1993*, pp. 83-92. Austin, Texas, USA. Society for Industrial and Applied Mathematics.

Ruppert, J. (1995) 'A Delaunay refinement algorithm for quality 2-dimensional mesh generation', *Journal of Algorithms*, 18(3), pp. 548-585.

Sadahiro, Y. (1997) 'Cluster perception in the distribution of point objects', *Cartographica*, 34(1), pp. 49-60.

Sarjakoski, L. T. (2007) 'Conceptual models of generalisation and multiple representation', in Mackaness, W.A., Ruas, A. and Sarjakoski, L.T. (eds.) *Generalisation of Geographic Information: Cartographic Modeling and Applications*. Amsterdam: Elsevier Science B.V., pp. 11-35.

Sarjakoski, L. T. and Nivala, A.-M. (2005) 'Adaptation to context - A way to improve the usability of mobile maps', *Map-based Mobile Services - Theories, Methods and Implementations*. Springer Berlin Heidelberg, pp. 107-123.

Savino, S. (2011) *A solution to the problem of the generalization of the Italian geographical databases from large to medium scale: approach definition, process design and operators implementation*. Doctoral dissertation, University of Padova, pages 141.

Schylberg, L. (1992) 'Cartographic amalgamation of area objects', In: *Proceedings of the ISPRS 1992*, pp. 135-138.

Seidel, R. (1991) 'A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons', *Computational Geometry*, 1(1), pp. 51-64. doi: [http://dx.doi.org/10.1016/0925-7721\(91\)90012-4](http://dx.doi.org/10.1016/0925-7721(91)90012-4).

Sester, M. (2000a) 'Generalization based on least squares adjustment. International archives of Photogrammetry and Remote Sensing', Vol. 33, Part B4, Amsterdam.

Sester, M. (2000b) 'Knowledge acquisition for the automatic interpretation of spatial data', *International Journal of Geographical Information Science*, 14(1), pp. 1-24. doi: 10.1080/136588100240930.

Sester, M. (2002) 'Application dependent generalization - The Case of Pedestrian Navigation', *Geospatial Theory, Processing and Applications*. Vol. 34/4, Ottawa, Canada.

Sharma, N., Bajapai, A. and Litoriya, R. (2012) 'Comparison the various clustering algorithms of WEKA tools', *International Journal of Emerging Technology and Advanced Engineering*, 2(5), pp. 73-80.

Shewchuk, J. (1999) 'Lecture notes on Delaunay mesh generation', in University of California at Berkeley.

Shewchuk, J. R. (1996) 'Triangle: Engineering a 2D quality mesh generator and Delaunay triangulators', In: *Proceedings of the first workshop on applied computational geometry*, pp. 124-133.

Sorrows, M. and Hirtle, M. (1999) 'The nature of landmarks for real and electronic spaces'. in Freska, C. and Mark, D. (eds.) *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*. Springer Verlag, pp. 37-50.

Steinhauer, J. H., Wiese, T., Freksa, C. and Barkowsky, T. (2001) 'Recognition of abstract regions in cartographic maps', in Montello, D.R. (ed.) *Spatial Information Theory*. Berlin Heidelberg New York: Springer, pp. 306-321.

Steiniger, S. and Weibel, R. (2005) 'A conceptual framework for automated generalization and its application to Geologic and Soil Maps', *22nd International Cartographic Association (ICC), a Coruña, Spain, 11-16th July 2005*.

Stoter, J., Burghardt, D., Duchêne, C., Baella, B., Bakker, N., Blok, C., Schmid, S. (2009) 'Methodology for evaluating automated map generalization in commercial software', *Computers, Environment and Urban Systems*, 33(5), pp. 311-324.

Stoter, J. E., Kraak, M. J. and Knippers, R. A. (2004) 'Generalization of framework data: a research agenda'. *International Cartographic Association (ICA) Workshop on Generalisation and Multiple representation, 20-21 August 2004, Leicester*.

- Stoter, J., E. (2005) 'Generalisation: The gap between research and practice'. In: *Proceedings of the 9th ICA workshop on generalisation and multiple representation*. A Coruña, Spain, 7–8 July 2005, pages 10.
- Su, B. O., Li, Z., Lodwick, G. and Muller, J.-C. (1997) 'Algebraic models for the aggregation of area features based upon morphological operators', *International Journal of Geographical Information Science*, 11(3), pp. 233-246. doi: 10.1080/136588197242374.
- Su, P. and Drysdale, R. L. S. (1995) 'A comparison of sequential Delaunay triangulation algorithms', In: *Proceedings of the 11th annual Symposium on Computational Geometry*. Vancouver, British Columbia, Canada. ACM.DOI: 10.1145/220279.220286.
- Su, P. and Scot Drysdale, R. L. (1997) 'A comparison of sequential Delaunay triangulation algorithms', *Computational Geometry*, 7(5-6), pp. 361-385.
- Thórisson, K. R. (1994) 'Simulated perceptual grouping: An application to Human-Computer interaction', In: *Proceedings of the Sixteenth annual conference of the cognitive science society*. Atlanta, Georgia, August 13-16. pp. 876-881.
- Timpf, S. (1999) 'Abstraction, levels of detail, and hierarchies in map series', in Stade, Germany, Freska, Christian and Mark and David, M. (eds.) *Spatial information theory - Cognitive and computational foundations of geographic information science*. Springer-Verlag, LNCS 1661, pp. 125-140.
- Toussaint, G. (1983) 'Solving geometric problems with rotating calipers', In: *Proceedings of Mediterranean Electrotechnical Conference (MELECON '83)*, Athens.
- Toussaint, G. T. (1991) 'Efficient triangulation of simple polygons', *Visual Computer*, 7, pp. 280-295.
- Tversky, B. and Lee, P. (1999) 'Pictorial and verbal tools for conveying routes'. *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*. Springer Berlin Heidelberg, pp. 51-64.
- Weka (2013) 'Weka 3: Data mining software in Java'. Available at: <http://www.cs.waikato.ac.nz/ml/weka/index.html> (Accessed 9th January 2013).
- Vivid Solutions JCS (2013) 'JCS conflation suite'. Available at: <http://www.vividsolutions.com/JCS/> (Accessed: 7th June 2013).
- Vivid Solutions JTS (2013) 'JTS topology suite'. Available at: <http://www.vividsolutions.com/jts/jtshome.htm> (Accessed: 7th July 2013).
- Vreda P. and Black, P.E. eds.(2004) 'Steiner point', in *Dictionary of Algorithms and Data Structures*. Available at: <http://www.nist.gov/dads/HTML/steinerpoint.html> (Accessed 18 January 2014).
- Wang, Z. and Müller, J. C. (1998) 'Line generalization based on analysis of shape characteristics', *Cartography and Geographic Information Systems*, 25(1), pp. 3-15.

- Ware, J., Jones, C., Bundy, G., Frank, A. and Kuhn, W. (1995) 'A triangulated spatial model for cartographic generalisation of areal objects: Spatial Information Theory A Theoretical Basis for GIS'. Springer Berlin / Heidelberg, pp. 173-192.
- Ware, J. M. and Jones, C. B. (1996) 'A spatial model for detecting (and resolving) conflict caused by scale reduction', *7th International Conference on Spatial data handling*. 1996, Delft. pp. 547-558.
- Ware, J. M., Jones, C. B. and Thomas, N. (2003) 'Automated map generalization with multiple operators: a simulated annealing approach', *International Journal of Geographical Information Science*, 17(8), pp. 743-769.
- Weibel, R. (1992) 'Models and experiments for adaptive computer-assisted terrain generalization', *Cartography and Geographic Information Science*, 19(3), pp. 133-153.
- Weibel, R. and Dutton, G. (1999) 'Generalizing spatial data and dealing with multiple representations'. in Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W. (eds.) *Geographical Information Systems: Principles and Technical issues*. New York: John Wiley and Sons, Second edition, pp. 125-155.
- Weibel, R. and Dutton, G. (1998) 'Constrained-based automated map generalization'. *International Symposium on Spatial data handling, Vancouver, Canada*, pp. 214-224.
- Weibel, R., Keller, S., Reichenbacher, T., Frank, A. and Kuhn, W. (1995) 'Overcoming the knowledge acquisition bottleneck in map generalization: The role of interactive systems and computational intelligence Spatial Information Theory A Theoretical Basis for GIS'. Springer Berlin / Heidelberg, pp. 139-156.
- Wertheimer, M. (1923) 'Laws of organization in perceptual forms'. *Psychologische Forschung*, 4, 301-350. Reprinted in part in Ellis, W.D. (ed.) *in a source book of Gestalt Psychology*. New York, pp. 71-88.
- Witten, H., Frank, E. and Hall, M. A. (2011) *Data mining: Practical machine learning tools and techniques*. Third Edition edn. Morgan Kaufmann, Burlington, USA.
- Zahn, C. T. (1971) 'Graph-theoretical methods for detecting and describing Gestalt clusters', *Computers, IEEE Transactions on*, C-20(1), pp. 68-86.
- Zhang, X., Ai, T. and Stoter, J. (2012) 'Characterization and detection of building patterns in cartographic data: Two Algorithms'. in Yeh, A.G.O., Shi, W., Leung, Y. and Zhou, C. (eds.), *Advances in Spatial Data Handling and GIS*. Springer Berlin Heidelberg, pp. 93-107.
- Zhang, X., Stoter, J., Ai, T. and Kraak, M.-J. (2012) 'Formalization and data enrichment for automated evaluation of building pattern preservation'. in Yeh, A.G.O., Shi, W., Leung, Y. and Zhou, C. (eds.), *Advances in Spatial Data Handling and GIS*. Springer Berlin Heidelberg, pp. 109-125.
- Zipf, A. and Jöst, M. (2006) 'Implementing adaptive mobile GI services based on ontologies: Examples from pedestrian navigation support', *Computers, Environment and Urban Systems*, 30(6), pp. 784-798.

Zipf, A. and Richter, K. (2002) 'Using focus maps to ease map reading - Developing smart applications for mobile devices', *KI 4/02 Special Issue Spatial Cognition*, 2, pp. 35-37.

Žalik, B. (2005) 'An efficient sweep-line Delaunay triangulation algorithm', *Computer-Aided Design*, 37(10), pp. 1027-1038. doi: <http://dx.doi.org/10.1016/j.cad.2004.10.004>.

Žalik, B. and Kolingerová, I. (2003) 'An incremental construction algorithm for Delaunay triangulation using the nearest-point paradigm', *International Journal of Geographical Information Science*, 17(2), pp. 119-138.

Appendices

A Prototypes of Graphical User Interfaces with proprietary software

This section provides a description of the prototypes developed in the form of GUIs using the proprietary Visual C# 2008 object-oriented programming language. These prototypes can connect to the open source PostgreSQL database coupled with the PostGIS extension (<http://postgis.refrations.net/>) to read building geometries stored therein using the open source NPGSQL data adaptor (<http://npgsql.projects.pgfoundry.org/>).

A.1 Input file creation for constrained Delaunay triangulation

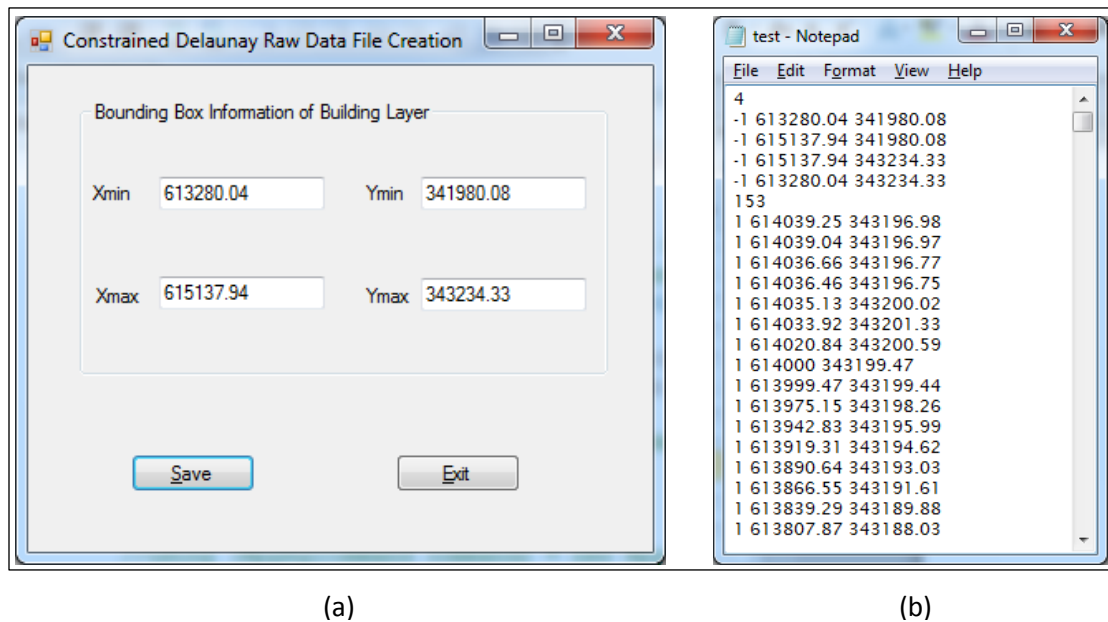


Figure A.1 (a) Dialogue menu with the coordinate values of the MBB of the building data set retrieved automatically for generating outer polygon of the input data structure and (b) input building geometries in ASCII format where polygon IDN of the outer polygon (MBB) is assigned -1.

A.2 Input file creation for the conforming Delaunay and the Delaunay constrained triangulations

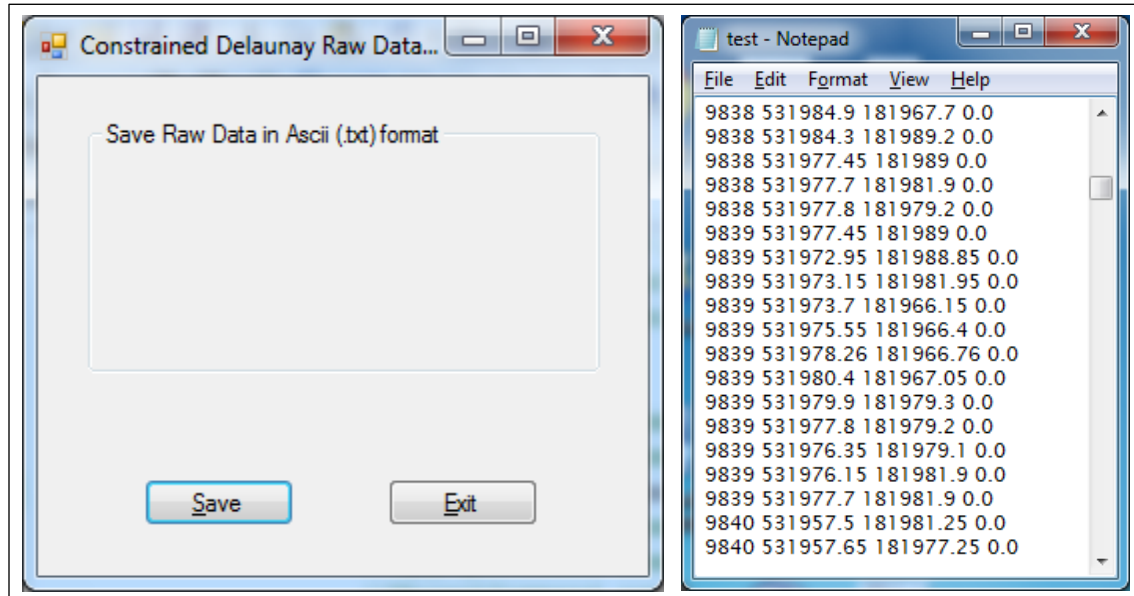


Figure A.2 (a) Dialogue menu to extract and save building polygon geometries in ASCII format and (b) extracted building outer polygon geometries with respective building IDNs.

B Prototypes of Graphical User Interfaces with open source software

This section provides a description of the prototypes developed in the form of GUIs using open source Java object-oriented programming language. Geographic features can be either stored in ASCII format or in the open source PostgreSQL database with the PostGIS extension (<http://postgis.refractory.net/>) to handle spatial data.

B.1 Constrained Delaunay triangulation

Once the constrained triangulation is executed with the GUI (Figure A.1 (a)), all building links of adjacency information with building IDNs [Building_IDN_From, Building_IDN_To] is written to a comma separated ASCII file (Figure A.1 (b)). These adjacency links are derived from the building IDNs attached to each triangle node as depicted in Figure 4.2, page 81.

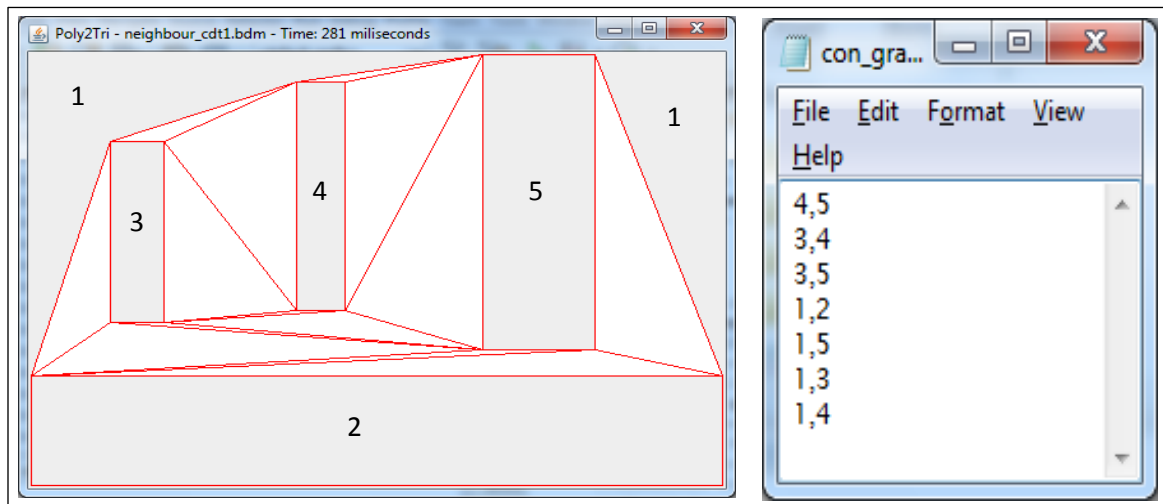
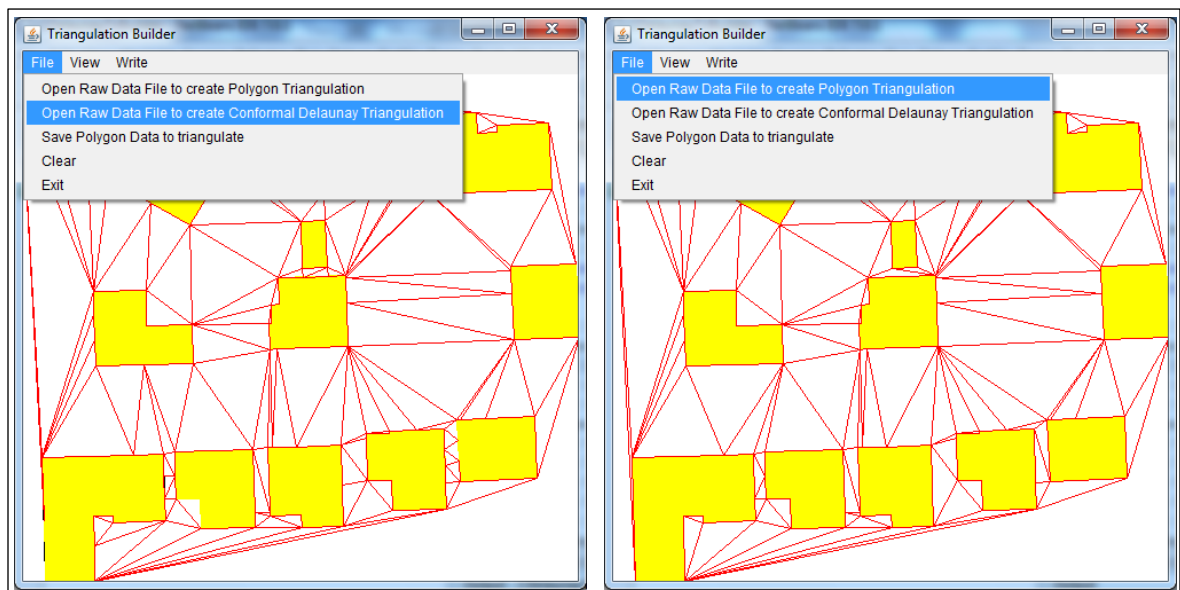


Figure B.1 (a) CDT output with rectangular buildings outlined in red colour and their IDNs on the GUI and (b) adjacency links between buildings where Polygon IDN 1 is the IDN of the outer polygon.

B.2 Conforming Delaunay triangulation and polygon triangulation

Both algorithms on CNDT and polygon triangulation have been implemented in the same GUI (Figure B.2) to read building geometries from ASCII files, view constrained triangulation result and write adjacency links between building geometries with building IDNs.



(a)

(b)

Figure B.2 (a) CNDT output and (b) output on polygon triangulation based algorithm with the edges of building polygons set as constraints, both implemented on the same GUI.

B.3 Constrained algorithm on Delaunay triangulation and spatial clustering

Algorithms on constrained triangulation, spatial clustering of building polygon geometries and the cluster shape enrichment have been implemented in the following GUI (Figure B.3). Data can be input into the GUI region-wise either in ASCII format or by directly reading data stored in the PostgreSQL database coupled with the PostGIS extension. The main functions of the prototype consist of generating triangulation in a particular region with the use of the regional IDN (known as the `global_id` field in the GUI), finding the Gestalt relations such as proximity, orientation difference and similarity difference in shape and size between building polygons with the adjacency information, creating building clusters, cluster shape enrichment and finally writing clustering results back to the spatial database.

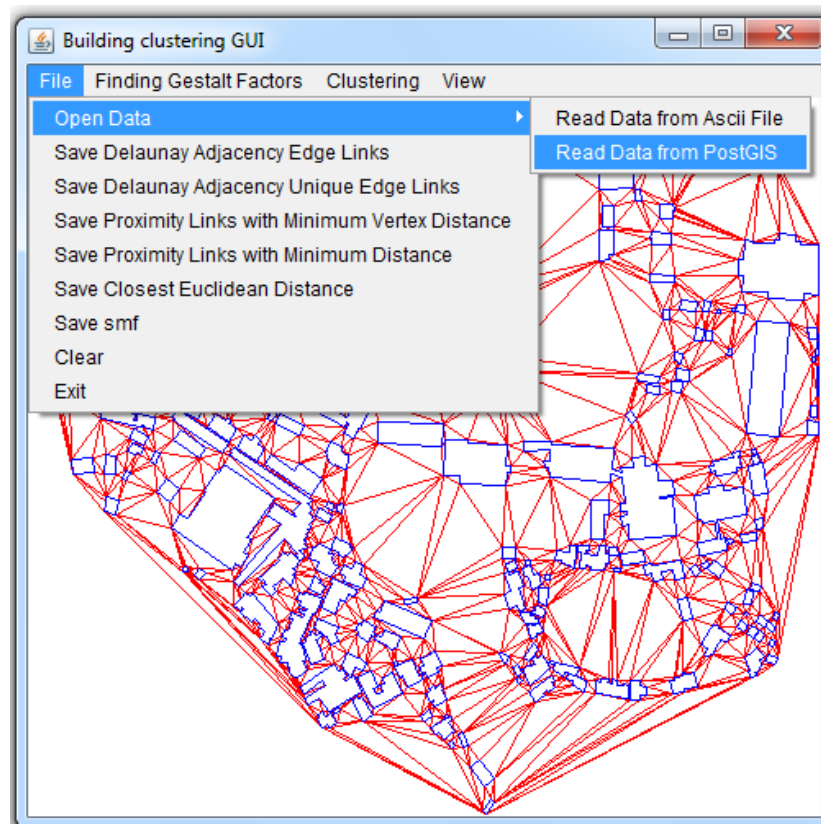


Figure B.3 GUI for spatial clustering of building polygon geometries in the data enrichment process with the use of constrained algorithm on Delaunay triangulation.

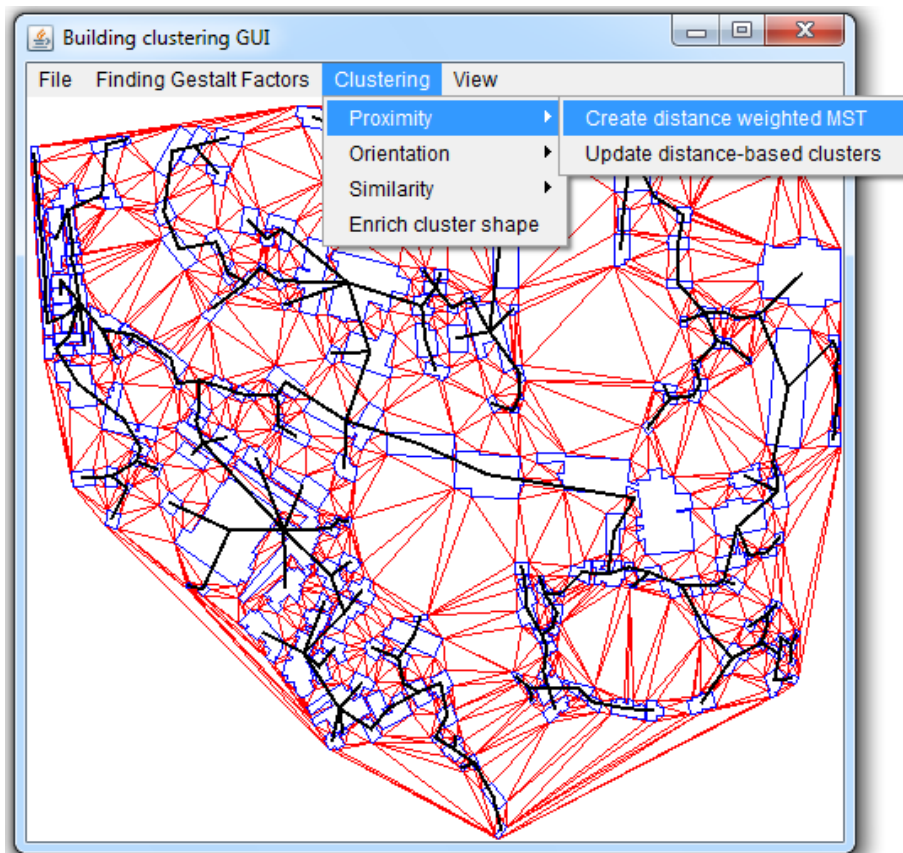


Figure B.4 Distance weighted initial MST in thick black lines used for clustering of building geometries.

B.4 Polygon cluster matching

This user interface writes cluster matching results in each region (partition) for each subject in both expert and lay groups. Clustering data is called from the PostGIS database, and the results are output in ASCII format and stored in tables (see Annex D.6).

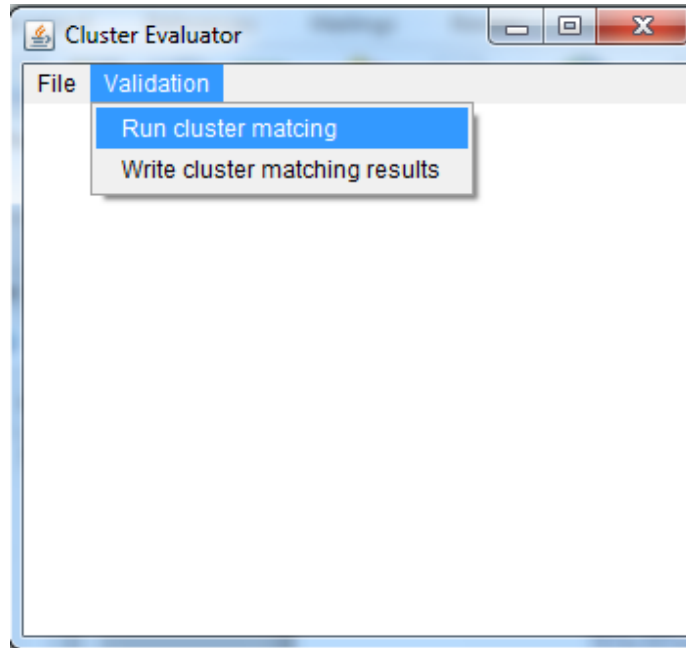


Figure B.5 UI for matching automatic clustering results with that of manual clustering by the subjects in both expert and lay groups.

B.5 Spatial clustering considering the context

Automatic clustering of building geometries, taking into consideration of context (contextual features: roads and hydrographic geometries) has been implemented with the use of the modified CNDT in the following GUI (Figure B.6). Further, the prototype is equipped with the functions to enhance cluster shape characteristics required for automatic map generalization and other geometric and spatial characteristics required in the data mining process for deriving salient building landmark geometries with the help of triangulation. Data can be input into the GUI region-wise either in ASCII format or by directly reading data stored in the PostgreSQL database coupled with the PostGIS extension. Finally, the enhanced results can be written back to the spatial database.

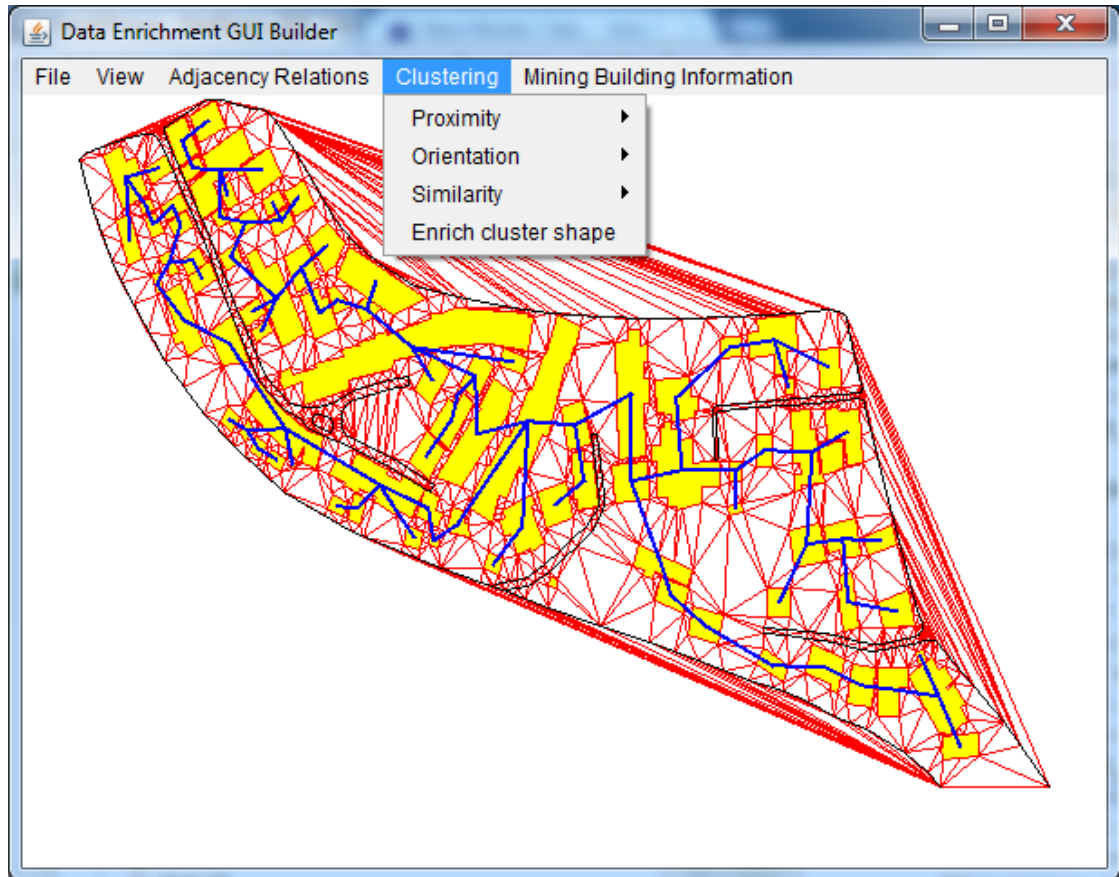


Figure B.6 GUI for spatial clustering of building polygon geometries with the consideration of the contextual features (roads in black colour) using the CNDT with edge constraints. Blue thick lines are the generated MST with the weight chosen as the adjacent distance of each pair of buildings.

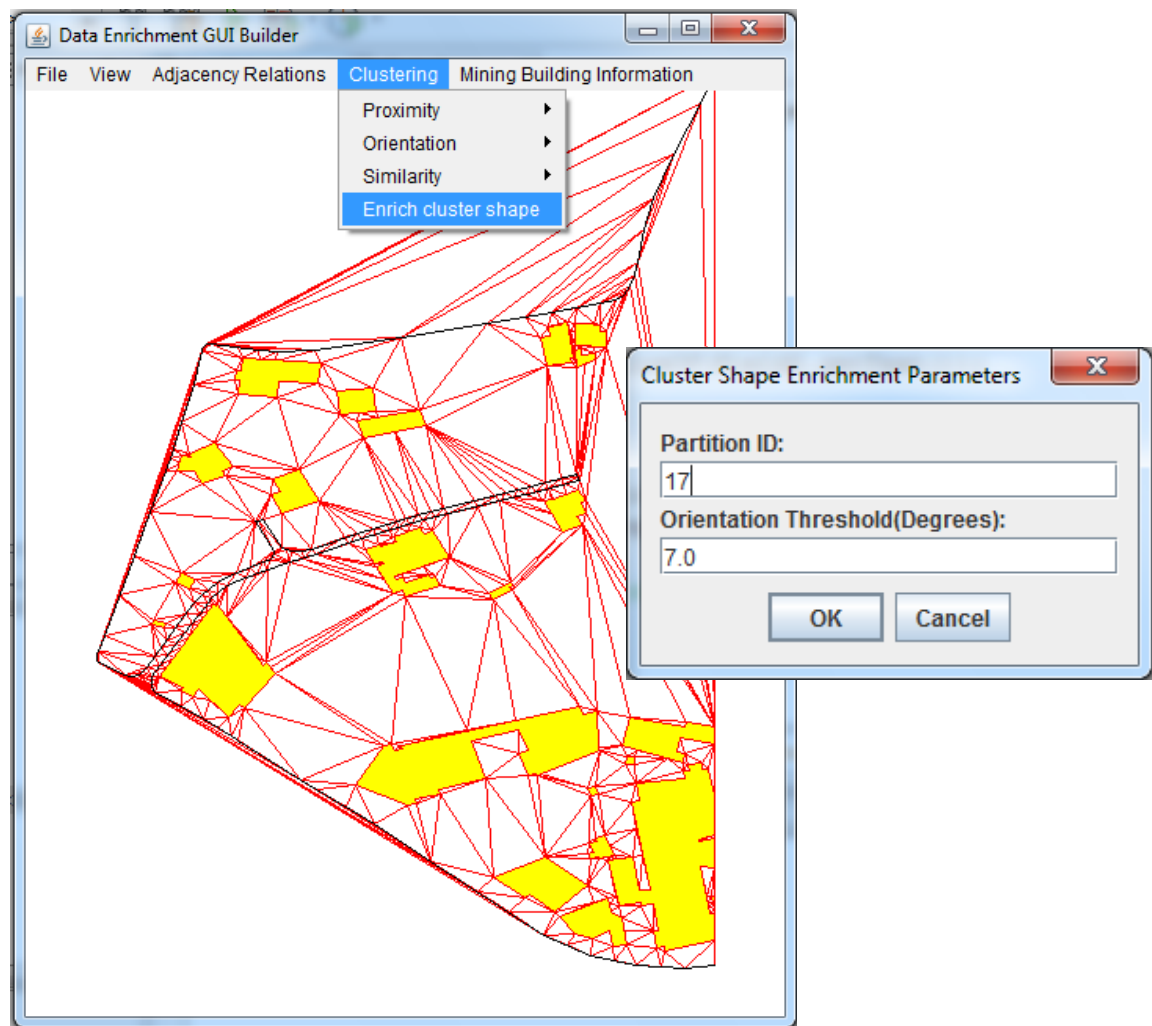


Figure B.7 Cluster shape enrichment of building clusters.

B.6 Generalization of building geometries

Algorithms for building aggregation developed in this research have been implemented in the following GUI (Figure B.8). Data can be input into the GUI region-wise either in ASCII format or by directly reading data stored in the PostgreSQL database coupled with the PostGIS extension. The main functions of the prototype consist of building polygon symbolization, aggregation, squaring, enlargement and simplification. The results can also be written back to the PostGIS database.

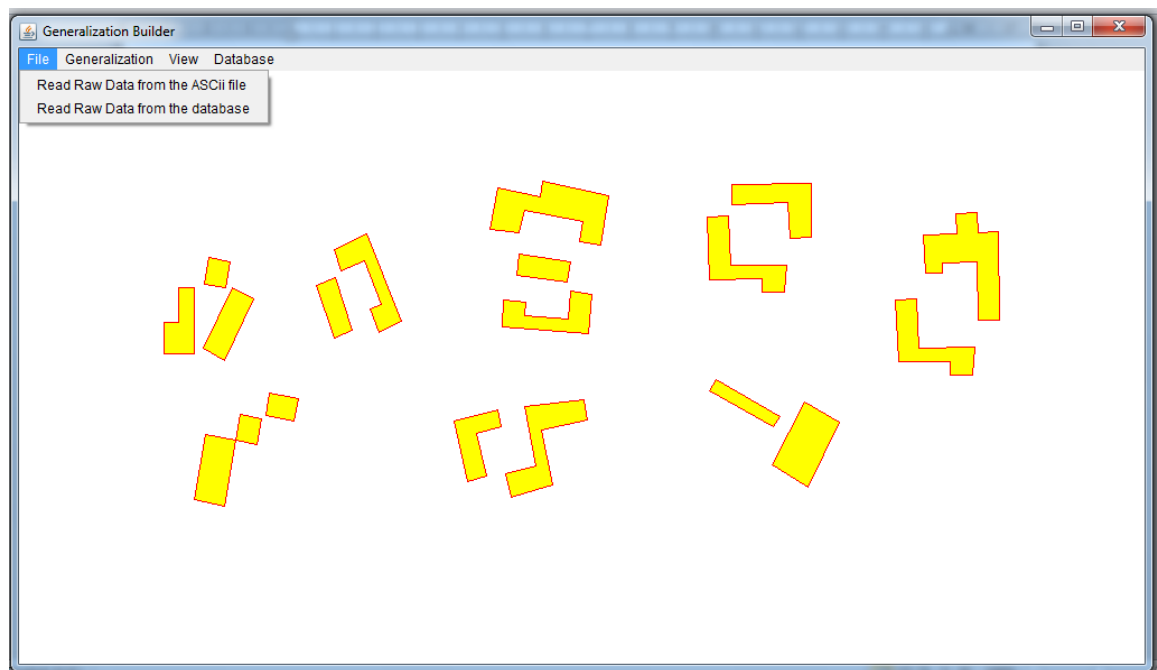


Figure B.8 Generalization GUI for building geometries.

C Existing algorithms

This section provides a description of the existing algorithms used in this research for implementing necessary tools and functions for generating focus maps.

C.1 Prim's algorithm for creating the Minimum Spanning Tree

Prim()

// cost[1..n][1..] is a cost adjacency matrix.

// n: number of vertices in the graph.

// cost [i][j] = ∞ , if there is no edge between i and j.

// cost [i][j] = ∞ , if i=j

// cost [i][j] = cost [j][i] = positive number if it is edge.

// t [1..n-1][1..2] has all the edges of minimum spanning tree

// mincost is the minimum cost

// near[] is an array which stores vertex in tree

// such that cost [j][near[j]] is minimum among all choices for near[j].

1. mincost =0;
2. For (i=2 to n) do near[i]=1;
3. Near[1]=0 //vertex 1 is initially in 't' .
4. For(i=1 to n-1) do
5. { // find n-1 edges of tree.
6. Let j be an index such that
7. Near[j] !=0 & cost[j][near[j]] is minimum.
8. // computation of j requires linear loop
9. // not shown above.
10. t[i][1] = j; t[i][2]= near[j];
11. mincost=mincost + cost[j][near [j]];


```
12. Near[ j ] =0;
13. For( k= 1 to n ) do //update near[ ]
14. If(near[ k ] !=0) && (cost[ k ][ near[ k ] ] > cost[ k ][ j ])
15. Near[ k ]=j;
16. }
17. return mincost.
```

C.2 Orientation of building polygons on the wall statistical weighting

- Select a series of candidate orientations between 0 and $\pi/2$ with a step depending on the required precision (e.g. 1 degree, i.e. $\pi/180$).
- For each candidate orientation, a weight is computed. This weight is the sum of all the edge contributions of a building. The contribution of an edge is computed as described in Figure C.1.
- The edge only contributes if its orientation is within a maximum deviation of δ from the candidate orientation and δ is a parameter empirically fixed at $\pi/12$. In this method, the orientation of an edge and the difference between orientations of the edges are considered modulo $\pi/2$. Thus an orientation of an edge not only contributes to the candidate orientations that are almost parallel to it, but also to the candidate orientations that are almost perpendicular to it (Figure C.1: edge (c) of the building contributes to candidate orientation α_i).
- Finally, get the wall statistical orientation which is the candidate orientation of the maximum weight.

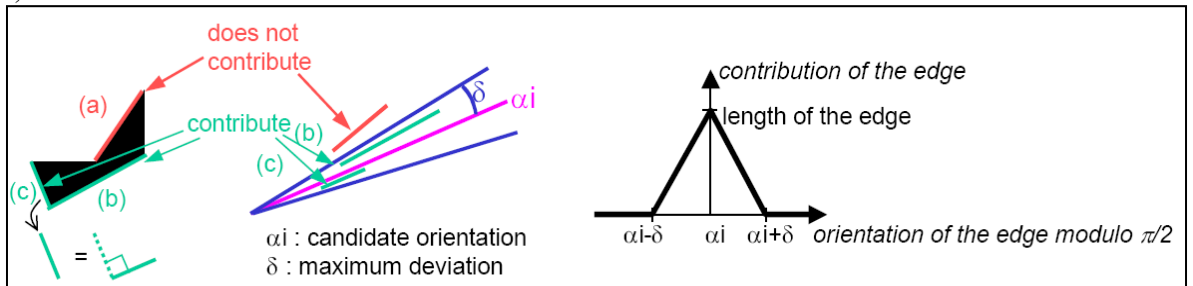


Figure C.1 Contribution of an edge of a building to a candidate orientation α_i on modulo $\pi/2$, from Duchêne *et al.* (2003).

C.3 Building simplification algorithm

- If the two connected edges are almost orthogonal, the shorter edge is removed, and the two other connected edges are extended until they intersect (Figure C.2(a)).
- If the two connected edges are almost parallel with the same orientation, the three edges are replaced with a single edge passing through the middle of the shorter edge with an average orientation (weighted by their length) of the two connected edges (Figure C.2(b)).
- If the two connected edges are almost parallel to the opposite orientations, it removes the three edges (Figure C.2(c)).
- After the edge removal operation, the validity of the geometry is checked. If it is not valid, the algorithm tries to delete another shorter edge, or else, it re-computes a new list of shorter edges and loop to the beginning. The algorithm stops if no shorter edges are present, or if there are no shorter edges to delete without breaking the validity of the geometry.

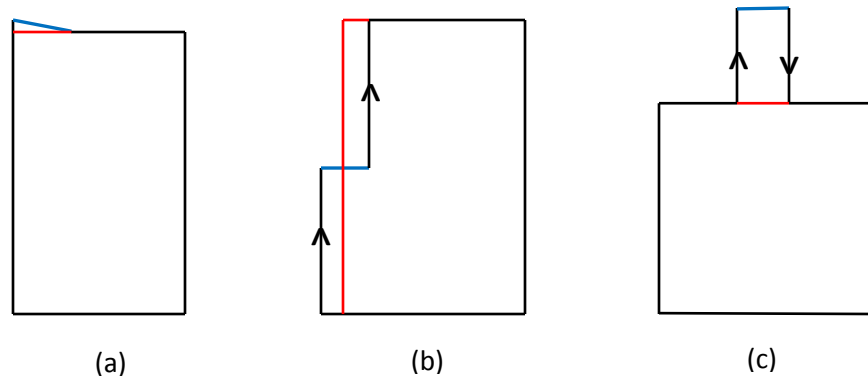


Figure C.2 Simplification of shorter edges: Two edges connecting a shorter edge in blue colour (a) almost perpendicular (b) almost parallel and (c) almost parallel to each other in the opposite direction. Edges in red colour are the replacements in the simplification.

D Clustering

This section provides resources and instructions given to the subjects regarding the clustering process in the user testing phase, and the subsequent data used to evaluate the building polygon clustering process.

D.1 Topographic maps used for the clustering experiment: Phase I

The topographic map (left) covering an area of 1Km x 0.75Km below represents building data and transport data in polygon and line formats respectively, printed at the scale of 1 : 4K from the 1 : 1K digital source data of the NMA of Sri Lanka. Right is the target map reduced and printed in 1 : 10K from the same digital source data with no generalization applied. These two printed maps on the original scales were given to each subject during the experimentation.

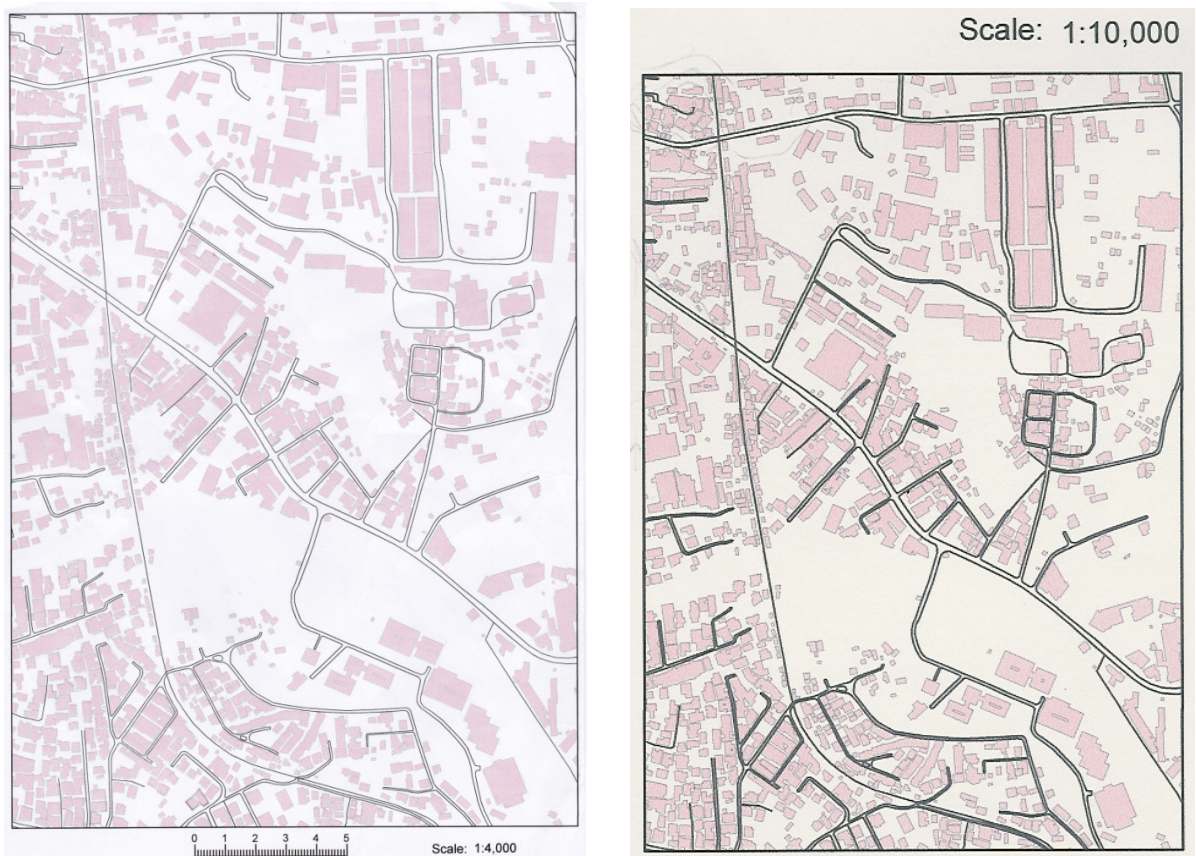


Figure D.1 Source map at 1 : 4K (left) and the target map reduced and printed at 1 : 10K (right), both not shown at the original scales.

D.2 Regions of the digital topographic data used for clustering



Figure D.2 Regions surrounded by the road network where the inner roads with dead ends are ignored.



Figure D.3 Building features in each region depicted by a unique colour.

D.3 Instruction to subjects in the clustering experiment: Phase I

Automatic hierarchical clustering approach with cluster classification adopted in this research was explained to the subjects in both groups in the beginning. Also, an idea about the map scale in order to measure ground distances of corresponding features on a map was given especially to the subjects in the lay group.

Clustering procedure to be adopted:

1. Identify the regions completely enclosed by the road network.
2. Consider a minimum separation distance between two objects to be 0.5mm (Note: see scale bar with 0.5mm graduation on the source map for your assistance).
3. Determine and write down distance threshold values (medium and very far) you use on the map in mm (as you perceive on the target map at the scale of 1 : 10K with the use of printed map at the reduced scale of 1 : 10K).
4. Write down the threshold values for orientation difference and similarity in shape and size.
5. Within each region given on the map in the consecutive order, perform clustering on your own, neglecting the inner roads with dead-ends and/or closed roads, bearing in mind the clustering hierarchy and the threshold values.
6. Delineate each cluster with pencil on the source map at the scale of 1 : 4K (Appendix D.1) and label them according to the cluster classification used in the automatic clustering.
7. Leave isolated buildings as single clusters on the source map.

D.4 Manual clustering output of a subject from the expert group



Figure D.4 Manual clustering output of a subject from the expert group.

D.5 Manual clustering output of a subject from the lay group



Figure D.5 Manual clustering output of a subject from the lay group.

D.6 Automatic and manual cluster matching results

Source clusters in Table D.1 are the clusters generated by the automatic clustering method, and the target clusters are the clusters created manually by the subjects. The classification of clusters (VC, ML, MS and MDS) is described in Chapter 5, Section 5.2.2. Classification label **M** denotes clusters of buildings in the medium range distance, labelled by the subjects.

Table D.1 Results of the expert group in each partitioned region.

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	1	2	1	0	0	0	0	0	1
E2		2	0	0	0	0	0	0	2
E3		2	1	0	0	0	0	0	1
E4		2	0	0	0	0	0	0	2
E5		2	1	0	0	0	0	0	1
E6		2	0	0	0	0	0	0	2
E7		2	0	0	0	0	0	0	2
E8		2	1	0	0	0	0	0	1
E9		2	1	0	0	0	0	0	1
E10		2	1	0	0	0	0	0	1
E11		2	1	0	0	0	0	0	1
E12		2	1	0	0	0	0	0	1
E13		2	0	0	0	0	0	0	2
E14		2	1	0	0	0	0	0	1
E15		2	0	0	0	0	0	0	2
E1	2	6	1	0	0	0	0	1	4
E2		6	0	0	0	0	0	1	5
E3		6	2	0	0	0	0	0	4
E4		6	0	0	0	0	0	1	5
E5		6	2	0	0	0	0	0	4
E6		6	1	0	0	0	0	1	4
E7		6	2	0	0	0	0	0	4
E8		6	1	0	0	0	0	0	5
E9		6	0	0	0	0	0	0	6
E10		6	0	0	0	0	0	2	4
E11		6	1	0	0	0	0	0	5
E12		6	2	0	0	0	0	0	4
E13		6	1	0	0	0	0	0	5
E14		6	2	0	0	0	0	1	3
E15		6	0	0	0	0	0	0	6

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	3	8	1	0	0	0	0	0	7
E2		8	2	0	0	0	1	2	3
E3		8	2	0	0	0	1	0	5
E4		8	1	0	0	0	1	0	6
E5		8	3	0	0	0	1	1	3
E6		8	0	0	0	0	1	2	5
E7		8	2	0	0	0	0	2	4
E8		8	1	0	0	0	0	0	7
E9		8	1	0	0	0	0	0	7
E10		8	2	0	0	0	0	0	6
E11		8	4	0	0	0	1	1	2
E12		8	3	0	0	0	0	0	5
E13		8	0	0	0	0	0	3	5
E14		8	0	0	0	0	0	4	4
E15		8	0	0	0	0	0	2	6
E1	4	9	1	0	0	0	0	0	8
E2		9	2	0	0	0	0	0	7
E3		9	4	0	0	0	0	0	5
E4		9	0	0	0	0	0	1	8
E5		9	3	0	0	0	0	1	5
E6		9	0	0	0	0	0	2	7
E7		9	2	0	0	0	0	1	6
E8		9	2	0	0	0	0	0	7
E9		9	0	0	0	0	0	1	8
E10		9	2	0	0	0	0	0	7
E11		9	4	0	0	0	0	0	5
E12		9	3	0	0	0	0	0	6
E13		9	1	0	0	0	0	0	8
E14		9	1	0	0	0	0	0	8
E15		9	0	0	0	0	0	1	8

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	5	52	3	0	0	0	0	0	49
E2		52	13	0	0	0	2	7	30
E3		52	8	0	0	0	0	1	43
E4		52	6	0	0	0	0	6	40
E5		52	21	0	0	0	1	1	29
E6		52	3	0	0	0	0	17	32
E7		52	6	0	0	0	0	7	39
E8		52	6	0	0	0	0	1	45
E9		52	2	0	0	0	0	1	49
E10		52	4	0	0	0	0	3	45
E11		52	15	0	0	0	0	15	22
E12		52	16	0	0	0	0	4	32
E13		52	0	0	0	0	0	7	45
E14		52	2	0	0	0	0	6	44
E15		52	4	0	0	0	0	3	45
E1	6	2	0	0	0	0	0	0	2
E2		2	2	0	0	0	0	0	0
E3		2	0	0	0	0	0	0	2
E4		2	1	0	0	0	0	0	1
E5		2	0	0	0	0	0	1	1
E6		2	0	0	0	0	0	0	2
E7		2	1	0	0	0	0	0	1
E8		2	0	0	0	0	0	0	2
E9		2	0	0	0	0	0	0	2
E10		2	0	0	0	0	0	0	2
E11		2	1	0	0	0	0	0	1
E12		2	1	0	0	0	0	1	0
E13		2	0	0	0	0	0	0	2
E14		2	0	0	0	0	0	0	2
E15		2	0	0	0	0	0	0	2

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	7	23	3	0	0	0	0	1	19
E2		23	7	0	0	0	0	4	12
E3		23	8	0	0	0	0	2	13
E4		23	2	0	0	0	0	3	18
E5		23	13	0	0	0	0	1	9
E6		23	2	0	0	0	0	11	10
E7		23	3	0	0	0	0	4	16
E8		23	2	0	0	0	0	0	21
E9		23	0	0	0	0	0	1	22
E10		23	3	0	0	0	0	1	19
E11		23	12	0	0	0	0	5	6
E12		23	9	0	0	0	0	2	12
E13		23	1	0	0	0	0	5	17
E14		23	4	0	0	0	0	0	19
E15		23	2	0	0	0	0	2	19
E1	8	1	0	0	0	0	0	1	0
E2		1	1	0	0	0	0	0	0
E3		1	1	0	0	0	0	0	0
E4		1	0	0	0	0	0	0	1
E5		1	1	0	0	0	0	0	0
E6		1	1	0	0	0	0	0	0
E7		1	0	0	0	0	0	0	1
E8		1	0	0	0	0	0	0	1
E9		1	1	0	0	0	0	0	0
E10		1	0	0	0	0	0	1	0
E11		1	1	0	0	0	0	0	0
E12		1	1	0	0	0	0	0	0
E13		1	0	0	0	0	0	1	0
E14		1	1	0	0	0	0	0	0
E15		1	1	0	0	0	0	0	0

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	9	8	1	0	0	0	0	0	7
E2		8	5	0	0	0	0	0	3
E3		8	5	0	0	0	0	0	3
E4		8	0	0	0	0	0	0	8
E5		8	5	0	0	0	0	0	3
E6		8	0	0	0	0	0	3	5
E7		8	3	0	0	0	0	0	5
E8		8	1	0	0	0	0	0	7
E9		8	0	0	0	0	0	0	8
E10		8	0	0	0	0	0	0	8
E11		8	5	0	0	0	0	0	3
E12		8	1	0	0	0	0	2	5
E13		8	1	0	0	0	0	1	6
E14		8	2	0	0	0	0	0	6
E15		8	1	0	0	0	0	2	5
E1	10	15	2	0	0	0	0	0	13
E2		15	2	0	0	0	0	4	9
E3		15	4	0	0	0	0	0	11
E4		15	1	0	0	0	0	2	12
E5		15	7	0	0	0	1	1	6
E6		15	2	0	0	0	0	5	8
E7		15	1	0	0	0	0	5	9
E8		15	2	0	0	0	0	0	13
E9		15	0	0	0	0	0	0	15
E10		15	5	0	0	0	0	3	7
E11		15	6	0	0	0	0	4	5
E12		15	6	0	0	0	0	1	8
E13		15	3	0	0	0	0	2	10
E14		15	3	0	0	0	0	1	11
E15		15	0	0	0	0	0	0	15

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	11	1	1	0	0	0	0	0	0
E2		1	0	0	0	0	0	0	1
E3		1	0	0	0	0	0	0	1
E4		1	0	0	0	0	0	0	1
E5		1	0	0	0	0	0	0	1
E6		1	1	0	0	0	0	0	0
E7		1	0	0	0	0	0	0	1
E8		1	1	0	0	0	0	0	0
E9		1	0	0	0	0	0	1	0
E10		1	0	0	0	0	0	0	1
E11		1	1	0	0	0	0	0	0
E12		1	0	0	0	0	0	0	1
E13		1	0	0	0	0	0	1	0
E14		1	1	0	0	0	0	0	0
E15		1	0	0	0	0	0	0	1
E1	12	1	0	0	0	0	0	0	1
E2		1	0	0	0	0	0	0	1
E3		1	0	0	0	0	0	0	1
E4		1	0	0	0	0	0	0	1
E5		1	0	0	0	0	0	0	1
E6		1	0	0	0	0	0	0	1
E7		1	0	0	0	0	0	0	1
E8		1	1	0	0	0	0	0	0
E9		1	0	0	0	0	0	1	0
E10		1	0	0	0	0	0	0	1
E11		1	1	0	0	0	0	0	0
E12		1	0	0	0	0	0	0	1
E13		1	0	0	0	0	0	0	1
E14		1	1	0	0	0	0	0	0
E15		1	0	0	0	0	0	0	1

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	13	2	1	0	0	0	0	0	1
E2		2	0	0	0	0	0	1	1
E3		2	1	0	0	0	0	0	1
E4		2	0	0	0	0	0	0	2
E5		2	1	0	0	0	0	0	1
E6		2	1	0	0	0	0	0	1
E7		2	1	0	0	0	0	0	1
E8		2	1	0	0	0	0	0	1
E9		2	0	0	0	0	0	1	1
E10		2	1	0	0	0	0	0	1
E11		2	1	0	0	0	0	0	1
E12		2	1	0	0	0	0	0	1
E13		2	1	0	0	0	0	0	1
E14		2	1	0	0	0	0	0	1
E15		2	0	0	0	0	0	1	1
E1	14	1	0	0	0	0	0	0	1
E2		1	1	0	0	0	0	0	0
E3		1	1	0	0	0	0	0	0
E4		1	0	0	0	0	0	0	1
E5		1	1	0	0	0	0	0	0
E6		1	1	0	0	0	0	0	0
E7		1	0	0	0	0	0	0	1
E8		1	1	0	0	0	0	0	0
E9		1	1	0	0	0	0	0	0
E10		1	0	0	0	0	0	0	1
E11		1	1	0	0	0	0	0	0
E12		1	1	0	0	0	0	0	0
E13		1	1	0	0	0	0	0	0
E14		1	1	0	0	0	0	0	0
E15		1	1	0	0	0	0	0	0

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	15	1	0	0	0	0	0	0	1
E2		1	1	0	0	0	0	0	0
E3		1	1	0	0	0	0	0	0
E4		1	0	0	0	0	0	0	1
E5		1	1	0	0	0	0	0	0
E6		1	1	0	0	0	0	0	0
E7		1	0	0	0	0	0	0	1
E8		1	1	0	0	0	0	0	0
E9		1	1	0	0	0	0	0	0
E10		1	0	0	0	0	0	0	1
E11		1	1	0	0	0	0	0	0
E12		1	1	0	0	0	0	0	0
E13		1	1	0	0	0	0	0	0
E14		1	1	0	0	0	0	0	0
E15		1	1	0	0	0	0	0	0
E1	16	3	0	0	0	0	0	0	3
E2		3	1	0	0	0	0	1	1
E3		3	0	0	0	0	0	0	3
E4		3	0	0	0	0	0	2	1
E5		3	1	0	0	0	0	0	2
E6		3	0	0	0	0	0	0	3
E7		3	0	0	0	0	0	1	2
E8		3	0	0	0	0	0	0	3
E9		3	0	0	0	0	0	0	3
E10		3	0	0	0	0	0	0	3
E11		3	1	0	0	0	0	1	1
E12		3	0	0	0	0	0	0	3
E13		3	0	0	0	0	0	0	3
E14		3	2	0	0	0	0	0	1
E15		3	1	0	0	0	0	0	2

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	17	6	1	0	0	0	0	0	5
E2		6	2	0	0	0	1	0	3
E3		6	2	0	0	0	0	0	4
E4		6	0	0	0	0	0	1	5
E5		6	3	0	0	0	0	0	3
E6		6	0	0	0	0	0	2	4
E7		6	0	0	0	0	0	1	5
E8		6	1	0	0	0	0	0	5
E9		6	0	0	0	0	0	0	6
E10		6	2	0	0	0	0	1	3
E11		6	2	0	0	0	0	1	3
E12		6	1	0	0	0	0	1	4
E13		6	0	0	0	0	0	2	4
E14		6	0	0	0	0	0	1	5
E15		6	0	0	0	0	0	1	5
E1	18	1	1	0	0	0	0	0	0
E2		1	0	0	0	0	0	0	1
E3		1	1	0	0	0	0	0	0
E4		1	0	0	0	0	0	0	1
E5		1	1	0	0	0	0	0	0
E6		1	1	0	0	0	0	0	0
E7		1	1	0	0	0	0	0	0
E8		1	1	0	0	0	0	0	0
E9		1	0	0	0	0	0	1	0
E10		1	0	0	0	0	0	1	0
E11		1	1	0	0	0	0	0	0
E12		1	1	0	0	0	0	0	0
E13		1	0	0	0	0	0	0	1
E14		1	1	0	0	0	0	0	0
E15		1	1	0	0	0	0	0	0

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	19	3	0	0	0	0	0	0	3
E2		3	0	0	0	0	1	1	1
E3		3	0	0	0	0	0	1	2
E4		3	0	0	0	0	0	0	3
E5		3	1	0	0	0	1	0	1
E6		3	0	0	0	0	1	0	2
E7		3	0	0	0	0	0	0	3
E8		3	1	0	0	0	0	1	1
E9		3	0	0	0	0	0	0	3
E10		3	0	0	0	0	0	0	3
E11		3	0	0	0	2	0	1	0
E12		3	0	0	0	0	0	1	2
E13		3	0	0	0	0	0	0	3
E14		3	1	0	0	0	0	0	2
E15		3	0	0	0	0	0	1	2
E1	20	6	1	0	0	0	0	0	5
E2		6	2	0	0	0	0	1	3
E3		6	3	0	0	0	0	0	3
E4		6	1	0	0	0	0	0	5
E5		6	4	0	0	0	0	1	1
E6		6	0	0	0	0	0	2	4
E7		6	0	0	0	0	0	0	6
E8		6	0	0	0	0	0	0	6
E9		6	0	0	0	0	0	0	6
E10		6	1	0	0	0	0	0	5
E11		6	2	0	0	0	0	3	1
E12		6	2	0	0	0	0	0	4
E13		6	0	0	0	0	0	1	5
E14		6	0	0	0	0	0	0	6
E15		6	0	0	0	0	0	0	6

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
E1	21	1	0	0	0	0	0	0	1
E2		1	0	0	0	0	0	0	1
E3		1	0	0	0	0	0	0	1
E4		1	0	0	0	0	0	0	1
E5		1	1	0	0	0	0	0	0
E6		1	0	0	0	0	0	1	0
E7		1	0	0	0	0	0	0	1
E8		1	1	0	0	0	0	0	0
E9		1	0	0	0	0	0	0	1
E10		1	0	0	0	0	0	0	1
E11		1	1	0	0	0	0	0	0
E12		1	1	0	0	0	0	0	0
E13		1	0	0	0	0	0	0	1
E14		1	1	0	0	0	0	0	0
E15		1	0	0	0	0	0	0	1
E1	22	4	1	0	0	0	0	0	3
E2		4	1	0	0	0	0	2	1
E3		4	2	0	0	0	0	0	2
E4		4	0	0	0	0	0	0	4
E5		4	0	0	0	0	0	0	4
E6		4	0	0	0	0	0	3	1
E7		4	1	0	0	0	0	0	3
E8		4	0	0	0	0	0	0	4
E9		4	0	0	0	0	0	0	4
E10		4	1	0	0	0	0	0	3
E11		4	3	0	0	0	0	0	1
E12		4	3	0	0	0	0	0	1
E13		4	2	0	0	0	0	0	2
E14		4	2	0	0	0	0	0	2
E15		4	0	0	0	0	0	1	3

Table D.2 Results of the lay group in each partitioned region.

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	1	2	1	0	0	0	0	0	1
L2		2	1	0	0	0	0	0	1
L3		2	1	0	0	0	0	0	1
L4		2	0	0	0	0	0	0	2
L5		2	1	0	0	0	0	0	1
L6		2	1	0	0	0	0	0	1
L7		2	1	0	0	0	0	0	1
L8		2	1	0	0	0	0	0	1
L9		2	1	0	0	0	0	0	1
L10		2	0	0	0	0	0	1	1
L11		2	0	0	0	0	0	0	2
L12		2	0	0	0	0	0	0	2
L13		2	1	0	0	0	0	0	1
L14		2	1	0	0	0	0	0	1
L15		2	0	0	0	0	0	1	1
L1	2	6	0	0	0	0	0	0	6
L2		6	1	0	0	0	0	0	5
L3		6	2	0	0	0	0	0	4
L4		6	0	0	0	0	0	0	6
L5		6	0	0	0	0	0	0	6
L6		6	1	0	0	0	0	0	5
L7		6	2	0	0	0	0	0	4
L8		6	0	0	0	0	0	1	5
L9		6	2	0	0	0	0	1	3
L10		6	0	0	0	0	0	1	5
L11		6	2	0	0	0	0	0	4
L12		6	0	0	0	0	0	0	6
L13		6	2	0	0	0	1	2	1
L14		6	1	0	0	0	1	1	3
L15		6	0	0	0	0	0	1	5

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	3	8	0	0	0	0	0	0	8
L2		8	4	0	0	0	0	0	4
L3		8	3	0	0	0	1	0	4
L4		8	1	0	0	0	0	0	7
L5		8	1	0	0	0	1	0	6
L6		8	1	0	0	0	0	1	6
L7		8	2	0	0	0	0	2	4
L8		8	0	0	0	0	0	3	5
L9		8	3	0	0	0	0	0	5
L10		8	0	0	0	0	0	0	8
L11		8	0	0	0	0	0	0	8
L12		8	0	0	0	0	0	0	8
L13		8	4	0	0	0	0	0	4
L14		8	0	0	0	0	0	0	8
L15		8	0	0	0	0	0	0	8
L1	4	9	2	0	0	0	0	0	7
L2		9	5	0	0	0	0	0	4
L3		9	3	0	0	0	1	0	5
L4		9	1	0	0	0	0	0	8
L5		9	1	0	0	0	0	0	8
L6		9	2	0	0	0	0	2	5
L7		9	1	0	0	0	0	1	7
L8		9	0	0	0	0	0	1	8
L9		9	3	0	0	0	0	0	6
L10		9	0	0	0	0	0	1	8
L11		9	2	0	0	0	0	0	7
L12		9	1	0	0	0	0	1	7
L13		9	2	0	0	0	0	0	7
L14		9	2	0	0	0	0	0	7
L15		9	1	0	0	0	0	0	8

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	5	52	0	0	0	0	0	3	49
L2		52	21	0	0	0	1	0	30
L3		52	12	0	0	0	0	1	39
L4		52	0	0	0	0	0	1	51
L5		52	6	0	0	0	0	1	45
L6		52	9	0	0	0	0	5	38
L7		52	4	0	0	0	0	1	47
L8		52	3	0	0	0	0	0	49
L9		52	2	0	0	0	0	2	48
L10		52	2	0	0	0	0	4	46
L11		52	5	0	0	0	0	2	45
L12		52	1	0	0	0	0	0	51
L13		52	15	0	0	0	2	1	34
L14		52	1	0	0	0	0	1	50
L15		52	0	0	0	0	0	1	51
L1	6	2	1	0	0	0	0	0	1
L2		2	1	0	0	0	0	0	1
L3		2	1	0	0	0	0	0	1
L4		2	0	0	0	0	0	1	1
L5		2	0	0	0	0	0	0	2
L6		2	0	0	0	0	0	0	2
L7		2	0	0	0	0	0	1	1
L8		2	0	0	0	0	0	0	2
L9		2	0	0	0	0	0	0	2
L10		2	1	0	0	0	0	0	1
L11		2	0	0	0	0	0	1	1
L12		2	0	0	0	0	0	0	2
L13		2	0	0	0	0	0	1	1
L14		2	0	0	0	0	0	0	2
L15		2	0	0	0	0	0	0	2

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	7	23	0	0	0	0	0	2	21
L2		23	10	0	0	0	1	0	12
L3		23	3	0	0	0	0	0	20
L4		23	1	0	0	0	0	0	22
L5		23	0	0	0	0	0	0	23
L6		23	5	0	0	0	0	0	18
L7		23	1	0	0	0	0	1	21
L8		23	5	0	0	0	0	1	17
L9		23	2	0	0	0	0	1	20
L10		23	1	0	0	0	0	4	18
L11		23	1	0	0	0	0	3	19
L12		23	1	0	0	0	0	0	22
L13		23	6	0	0	0	0	1	16
L14		23	1	0	0	0	0	0	22
L15		23	0	0	0	0	0	0	23
L1	8	1	0	0	0	0	0	1	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	0	0	0	0	0	1	0
L5		1	1	0	0	0	0	0	0
L6		1	0	0	0	0	0	1	0
L7		1	0	0	0	0	0	1	0
L8		1	0	0	0	0	0	0	1
L9		1	1	0	0	0	0	0	0
L10		1	1	0	0	0	0	0	0
L11		1	0	0	0	0	0	1	0
L12		1	0	0	0	0	0	1	0
L13		1	0	0	0	0	0	1	0
L14		1	0	0	0	0	0	1	0
L15		1	0	0	0	0	0	1	0

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	9	8	1	0	0	0	0	0	7
L2		8	6	0	0	0	0	0	2
L3		8	1	0	0	0	0	0	7
L4		8	2	0	0	0	0	0	6
L5		8	0	0	0	0	0	0	8
L6		8	0	0	0	0	0	0	8
L7		8	1	0	0	0	0	1	6
L8		8	3	0	0	0	0	0	5
L9		8	3	0	0	0	0	0	5
L10		8	0	0	0	0	0	3	5
L11		8	2	0	0	0	0	0	6
L12		8	0	0	0	0	0	0	8
L13		8	6	0	0	0	0	0	2
L14		8	1	0	0	0	0	0	7
L15		8	0	0	0	0	0	0	8
L1	10	15	0	0	0	0	0	0	15
L2		15	10	0	0	0	0	0	5
L3		15	0	0	0	0	0	0	15
L4		15	2	0	0	0	0	2	11
L5		15	0	0	0	0	0	1	14
L6		15	5	0	0	0	0	1	9
L7		15	3	0	0	0	0	2	10
L8		15	0	0	0	0	0	1	14
L9		15	2	0	0	0	0	0	13
L10		15	0	0	0	0	0	0	15
L11		15	0	0	0	0	0	0	15
L12		15	0	0	0	0	0	1	14
L13		15	2	0	0	0	1	1	11
L14		15	0	0	0	0	0	0	15
L15		15	0	0	0	0	0	1	14

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	11	1	0	0	0	0	0	1	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	0	0	0	0	0	0	1
L5		1	0	0	0	0	0	0	1
L6		1	0	0	0	0	0	0	1
L7		1	1	0	0	0	0	0	0
L8		1	0	0	0	0	0	0	1
L9		1	0	0	0	0	0	0	1
L10		1	1	0	0	0	0	0	0
L11		1	1	0	0	0	0	0	0
L12		1	0	0	0	0	0	1	0
L13		1	0	0	0	0	0	0	1
L14		1	0	0	0	0	0	0	1
L15		1	0	0	0	0	0	0	1
L1	12	1	0	0	0	0	0	1	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	0	0	0	0	0	0	1
L5		1	0	0	0	0	0	0	1
L6		1	0	0	0	0	0	0	1
L7		1	0	0	0	0	0	0	1
L8		1	0	0	0	0	0	0	1
L9		1	0	0	0	0	0	0	1
L10		1	0	0	0	0	0	0	1
L11		1	0	0	0	0	0	1	0
L12		1	0	0	0	0	0	1	0
L13		1	0	0	0	0	0	0	1
L14		1	0	0	0	0	0	0	1
L15		1	0	0	0	0	0	0	1

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	13	2	0	0	0	0	0	1	1
L2		2	1	0	0	0	0	0	1
L3		2	1	0	0	0	0	0	1
L4		2	0	0	0	0	0	0	2
L5		2	0	0	0	0	0	0	2
L6		2	0	0	0	0	1	0	1
L7		2	0	0	0	0	0	0	2
L8		2	1	0	0	0	0	0	1
L9		2	0	0	0	0	0	0	2
L10		2	0	0	0	0	0	1	1
L11		2	1	0	0	0	0	1	0
L12		2	0	0	0	0	0	1	1
L13		2	0	0	0	0	0	1	1
L14		2	0	0	0	0	0	0	2
L15		2	0	0	0	0	0	1	1
L1	14	1	1	0	0	0	0	0	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	1	0	0	0	0	0	0
L5		1	1	0	0	0	0	0	0
L6		1	1	0	0	0	0	0	0
L7		1	1	0	0	0	0	0	0
L8		1	1	0	0	0	0	0	0
L9		1	1	0	0	0	0	0	0
L10		1	1	0	0	0	0	0	0
L11		1	1	0	0	0	0	0	0
L12		1	1	0	0	0	0	0	0
L13		1	1	0	0	0	0	0	0
L14		1	1	0	0	0	0	0	0
L15		1	1	0	0	0	0	0	0

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	15	1	1	0	0	0	0	0	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	1	0	0	0	0	0	0
L5		1	1	0	0	0	0	0	0
L6		1	1	0	0	0	0	0	0
L7		1	0	0	0	0	0	1	0
L8		1	1	0	0	0	0	0	0
L9		1	1	0	0	0	0	0	0
L10		1	1	0	0	0	0	0	0
L11		1	1	0	0	0	0	0	0
L12		1	1	0	0	0	0	0	0
L13		1	1	0	0	0	0	0	0
L14		1	1	0	0	0	0	0	0
L15		1	0	0	0	0	0	0	1
L1	16	3	0	0	0	0	0	0	3
L2		3	0	0	0	0	0	0	3
L3		3	0	0	0	0	0	2	1
L4		3	0	0	0	0	0	0	3
L5		3	0	0	0	0	0	0	3
L6		3	0	0	0	0	0	2	1
L7		3	0	0	0	0	0	0	3
L8		3	0	0	0	0	0	0	3
L9		3	2	0	0	0	0	0	1
L10		3	0	0	0	0	0	0	3
L11		3	0	0	0	0	0	0	3
L12		3	0	0	0	0	0	0	3
L13		3	0	0	0	0	0	0	3
L14		3	0	0	0	0	0	0	3
L15		3	0	0	0	0	0	0	3

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	17	6	0	0	0	0	0	0	6
L2		6	2	0	0	0	0	0	4
L3		6	1	0	0	0	0	0	5
L4		6	0	0	0	0	0	0	6
L5		6	0	0	0	0	0	0	6
L6		6	0	0	0	0	0	2	4
L7		6	1	0	0	0	0	0	5
L8		6	0	0	0	0	0	0	6
L9		6	0	0	0	0	0	0	6
L10		6	0	0	0	0	0	2	4
L11		6	1	0	0	0	0	1	4
L12		6	0	0	0	0	0	0	6
L13		6	1	0	0	0	0	2	3
L14		6	0	0	0	0	0	0	6
L15		6	0	0	0	0	0	0	6
L1	18	1	0	0	0	0	0	0	1
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	0	0	0	0	0	0	1
L5		1	0	0	0	0	0	0	1
L6		1	1	0	0	0	0	0	0
L7		1	0	0	0	0	0	0	1
L8		1	0	0	0	0	0	0	1
L9		1	0	0	0	0	0	0	1
L10		1	0	0	0	0	0	0	1
L11		1	0	0	0	0	0	0	1
L12		1	0	0	0	0	0	0	1
L13		1	1	0	0	0	0	0	0
L14		1	0	0	0	0	0	0	1
L15		1	0	0	0	0	0	0	1

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	19	3	0	0	0	0	0	0	3
L2		3	0	0	0	0	0	0	3
L3		3	0	0	0	0	1	0	2
L4		3	0	0	0	0	1	0	2
L5		3	0	0	0	0	0	0	3
L6		3	0	0	0	0	0	0	3
L7		3	0	0	0	0	0	1	2
L8		3	0	0	0	0	0	0	3
L9		3	0	0	0	0	0	0	3
L10		3	0	0	0	0	0	0	3
L11		3	0	0	0	0	0	0	3
L12		3	0	0	0	0	0	0	3
L13		3	0	0	0	0	0	0	3
L14		3	0	0	0	0	2	0	1
L15		3	0	0	0	0	1	0	2
L1	20	6	0	0	0	0	0	0	6
L2		6	2	0	0	0	0	0	4
L3		6	0	0	0	0	0	0	6
L4		6	0	0	0	0	0	1	5
L5		6	1	0	0	0	0	0	5
L6		6	1	0	0	0	0	1	4
L7		6	0	0	0	0	0	0	6
L8		6	1	0	0	0	0	0	5
L9		6	0	0	0	0	0	0	6
L10		6	1	0	0	0	0	0	5
L11		6	0	0	0	0	0	1	5
L12		6	0	0	0	0	0	0	6
L13		6	0	0	0	0	0	1	5
L14		6	0	0	0	0	0	0	6
L15		6	0	0	0	0	0	0	6

Subject	Region IDN	No. of source clusters	No. of correctly classified target clusters with source clusters					No. of source clusters matched but misclassified	No. of unmatched source clusters
			VC	ML	MS	MDS	M		
L1	21	1	0	0	0	0	1	0	0
L2		1	1	0	0	0	0	0	0
L3		1	1	0	0	0	0	0	0
L4		1	0	0	0	0	0	0	1
L5		1	0	0	0	0	0	0	1
L6		1	0	0	0	0	0	0	1
L7		1	0	0	0	0	0	1	0
L8		1	0	0	0	0	0	0	1
L9		1	0	0	0	0	0	0	1
L10		1	0	0	0	0	0	0	1
L11		1	0	0	0	0	0	0	1
L12		1	0	0	0	0	0	0	1
L13		1	0	0	0	0	0	0	1
L14		1	0	0	0	0	0	0	1
L15		1	0	0	0	0	0	0	0
L1	22	4	0	0	0	0	0	1	3
L2		4	2	0	0	0	0	0	2
L3		4	0	0	0	0	0	0	4
L4		4	1	0	0	0	0	0	3
L5		4	0	0	0	0	0	0	4
L6		4	3	0	0	0	0	0	1
L7		4	0	0	0	0	0	1	3
L8		4	2	0	0	0	0	0	2
L9		4	0	0	0	0	0	0	4
L10		4	0	0	0	0	0	1	3
L11		4	2	0	0	0	0	0	2
L12		4	1	0	0	0	0	0	3
L13		4	3	0	0	0	0	0	1
L14		4	0	0	0	0	0	0	4
L15		4	0	0	0	0	0	0	4

D.7 Questionnaire for evaluating the automatic clustering

USER SATISFACTION SURVEY on the RESULTS of AUTOMATIC CLUSTERING of BUILDING FEATURES to be COMPARED with the MANUAL CLUSTERING OUTPUT

I. Introduction

Dear participant,

The automatic clustering approach you have already known is dedicated to improving quality of map generalization, enabling the application of the right choice of generalization operations such as selection, removal, aggregation, simplification, enlargement, etc. This brief survey under Phase - II, which is a continuation of the Phase – I, is to gather your vital answers based on the comparison you made between the results obtained from the automatic clustering and your manual clustering output of the same data at Phase - I based on the same clustering algorithm. Your answers will be helpful in evaluating the algorithm further to achieve promising results in automatic map generalization. Your response will only be used for survey purposes. Attached is your manual clustering results on the 1 : 4K topographic map you performed in Phase - I of the experiment together with the same data printed at the scale of 1 : 10K; the target scale on which data is to be produced by generalization. The results of the automatic clustering will be presented to you at the beginning of the survey. Thank you very much for your time and suggestions.

Note: The results of the automatic clustering for producing a generalized map at the target scale 1 : 10K is based on the following threshold values for the three (03) criteria used hierarchically - distance, orientation difference and similarity difference in shape - between building features.

Distance range:

Very Close (dc): $vc \leq 0.5\text{mm}$ on map

Medium distance range (md): $2.0\text{mm} \geq md > 0.5\text{mm}$

Very Far (vf): $vf > 2.0\text{mm}$

Orientation difference (od): 7^0 . If $od > 7^0$ (degrees), orientation difference is considered to be large, or else is considered to be small.

Similarity difference in shape (sd) <0 – very similar, 1 – dissimilar>: If $sd \leq 0.25$, a pair of buildings is considered to be similar in shape and if $sd > 0.25$, a pair of buildings is considered to be dissimilar.

II. Questions

Directions: Please indicate your level of agreement or disagreement with each of these statements regarding automated and manual clustering approach. Place an “X” mark in the box for your answer.

Q1: How do you rate the output of the automated clustering algorithm?

Excellent ☐ *Good* ☐ *Fair* ☐ *Poor* ☐

Q2: Do you think that the classification used in identifying clusters is very much useful for subsequent map generalization?

Strongly agree ☐ *Agree* ☐ *Disagree* ☐ *Strongly Disagree* ☐

Q3: When comparing automated results with your results of the manual clustering approach, do you observe that the both results comply each other?

Strongly match ☐ *Match* ☐ *Mismatch* ☐ *Strongly mismatch* ☐

(Please give reasons)

Q4: How do you feel about the application of the algorithm to cluster building polygons manually with the hierarchical perception of Gestalt constraints; distance, orientation and similarity in shape?

Very easy ☐ Easy ☐ Difficult ☐ Very difficult ☐

(Please give reasons)

Q5: To what extent do you agree that the distance threshold values used for medium range in the automated clustering (i.e. between 0.5mm – 2.0mm on map) and very far value (i.e. > 2mm on map) comply with the range you used for manual clustering application of the algorithm when analysing your output with the automated result?

Strongly agree ☐ Agree ☐ Disagree ☐ Strongly disagree ☐

(Please write down your own threshold range/value, if your values do not agree with the values used in automated clustering above)

Q6: Do you agree with the threshold value (i.e. 7° degrees) used to distinguish the difference in orientation between a neighbouring pair of buildings when observing the results of the automated clustering?

Yes ☐ No ☐

If your answer is 'No', which of the following values in degrees you perceive to be the best?

3.5° ☐ 10° ☐ 14° ☐ 7.5° ☐

(If any other value, please specify)

Q7: Do you agree with the threshold value (i.e. 0.25) used to distinguish the similarity in shape between a neighbouring pair of buildings when observing the results of the automated clustering?

Yes ☐ No ☐

If your answer is 'No', which of the following values in degrees you perceive to be the best?

0.1 ☐ 0.15 ☐ 0.3 ☐ 0.4 ☐

(If any other value, please specify)

Q8: When analysing your manual clustering results with the automated results, do you feel that the attempt you made in manual clustering in the Phase I had the general idea of hierarchical clustering?

Yes ☐ No ☐

Q9: Do you find a significant difference between your cluster perception and the automated clustering when comparing your manual results with the automated results?

Yes ☐ No ☐

(Please give reasons, if your answer is 'Yes')

Q10: How would you rate your overall experience in the hierarchical clustering approach?

Satisfactory ☐ Neutral ☐ Unsatisfactory ☐ Highly unsatisfactory ☐

III. DEMOGRAPHIC DATA

Name (optional): _____

Age: 18 -30 ☐ 31-40 ☐ 41-54 ☐ 55 and over ☐

Sex: male ☐ female ☐

Email Address (optional): _____

IV. CLOSE

Thank you for sharing your thoughts in this survey. Enjoy in the field of research.

E SQL Queries

E.1 Handling PostGIS geometries in the PostgreSQL database

(1) Retrieve Bounding Box of a Polygon data set

```
SELECT gid, ST_XMin(BOX2D((ST_Dump(the_geom)).geom)) as XMin,  
  
ST_YMin(BOX2D((ST_Dump(the_geom)).geom)) as YMin,  
  
ST_XMax(BOX2D((ST_Dump(the_geom)).geom)) as XMAX,  
  
ST_YMax(BOX2D((ST_Dump(the_geom)).geom)) as YMAX  
  
from building_poly;
```

(2) Retrieving PostGIS geometry of each building polygon in the Well-Known Text (WKT) representation of the geometry/geography without SRID metadata where gid field represents the building feature IDN:

```
NpgsqlCommand command5 = new NpgsqlCommand("select  
gid,ST_AsText(St_geometryn(the_geom,1)) from topo_area where gid  
<= 14", conn);
```

(3) Obtaining union geometry within a region defined by glblId = 1 without null and non-single clusters

```
SELECT ST_Union(the_geom),cluster_id FROM experiment.rpoly  
  
WHERE cluster_id is not null and cluster_id NOT LIKE '%-0' and  
glbl_id = 1  
  
GROUP BY cluster_id;
```

(4) Finding neighbourhood density around a building with a buffer

```
/*
This query is to calculate and write the immediate neighbourhood
density of each building within a radius of 5m from the centre of
each building.
Result is written to an attribute field called "neigh_density"
with decimal places rounding to four
*/
UPDATE my_schema.testbuildings SET neigh_density = round(t1.nv,4)
from
(SELECT a.gid as id, count(*)/(3.14159 * 100 * 100) as nv
FROM my_schema.testbuildings a, my_schema.testbuildings b
WHERE ST_DWithin(ST_Centroid(a.the_geom), b.the_geom, 100)
GROUP BY a.gid) t1
WHERE t1.id = my_schema.testbuildings.gid;
```

(5) Creating a table in PostgreSQL with PostGIS geometry handling to store generalized results

```
/*
This query is to create a table with the geometry column to handle
geometry polygons (two-dimensional) in PostGIS 2.0
*/

CREATE TABLE generalized_polys (

    gid serial NOT NULL,

    PRIMARY KEY(gid),

    cluster_id varchar(12)

);

SELECT AddGeometryColumn('generalized_polys', 'the_geom', 0,
'POLYGON', 2 );
```

(6) Creating ConcaveHull with PostGIS 2.0 on building clusters

```
SELECT (ST_cleangeometry((ST_ConcaveHull(ST_Collect(the_geom),
    0.99)))), cluster_id
FROM concavepoly
GROUP BY cluster_id;
```

(7) Creating a union of each building cluster of orthogonal shape characteristics within a region (glbl_id = 1) with PostGIS 2.0

```
SELECT ST_Union(the_geom), cluster_id FROM experiment.rpoly
where glbl_id = 1 AND cluster_shape = 1 AND cluster_id NOT LIKE '%-0'
GROUP BY cluster_id;
```

(8) Creating UnaryUnion of adjoining buildings in a cluster with non-orthogonal shape characteristics within a region (glbl_id = 1) with PostGIS 2.0

```
SELECT ST_UnaryUnion(ST_Collect(the_geom)), cluster_id FROM
experiment.rpoly
where glbl_id = 1 AND cluster_shape = 0 AND cluster_id NOT LIKE '%-0'
GROUP BY cluster_id;
```

F Data mining

F.1 Sensitivity analysis workflow in the WEKA software

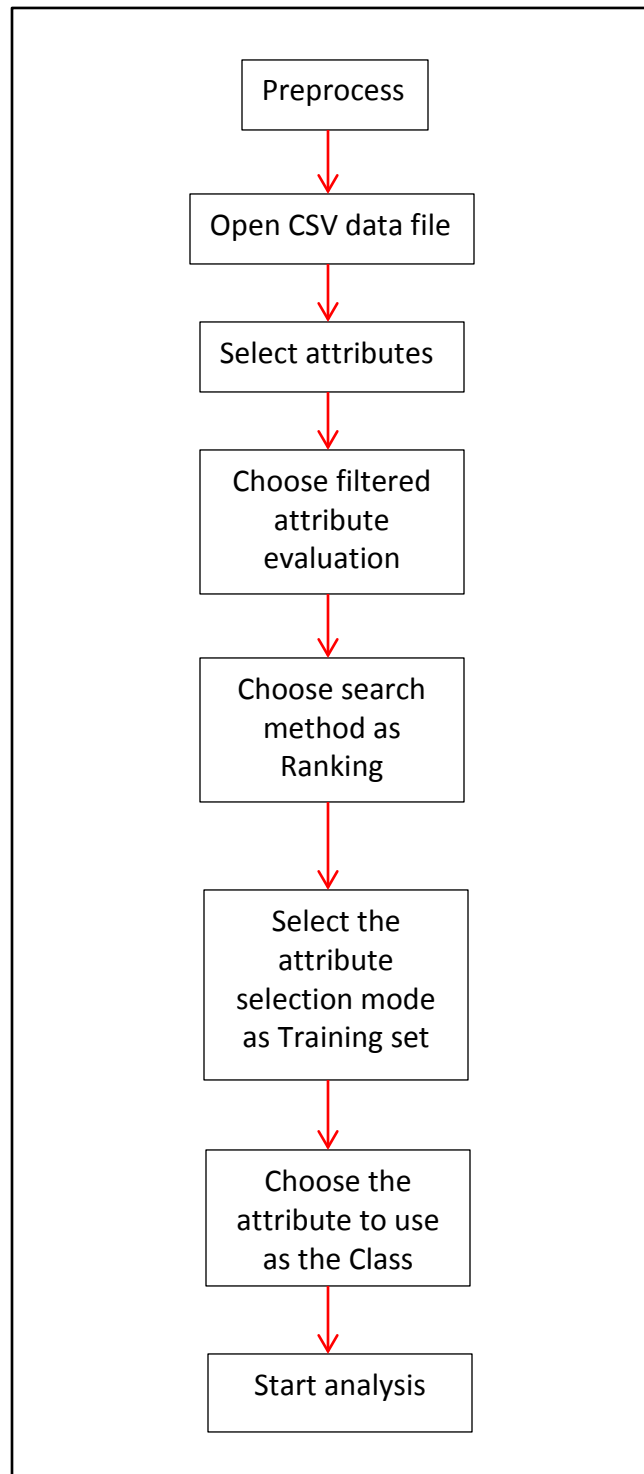


Figure F.1 Sensitivity analysis workflow in the WEKA GUI.

F.2 Results of the sensitivity analysis in the WEKA software

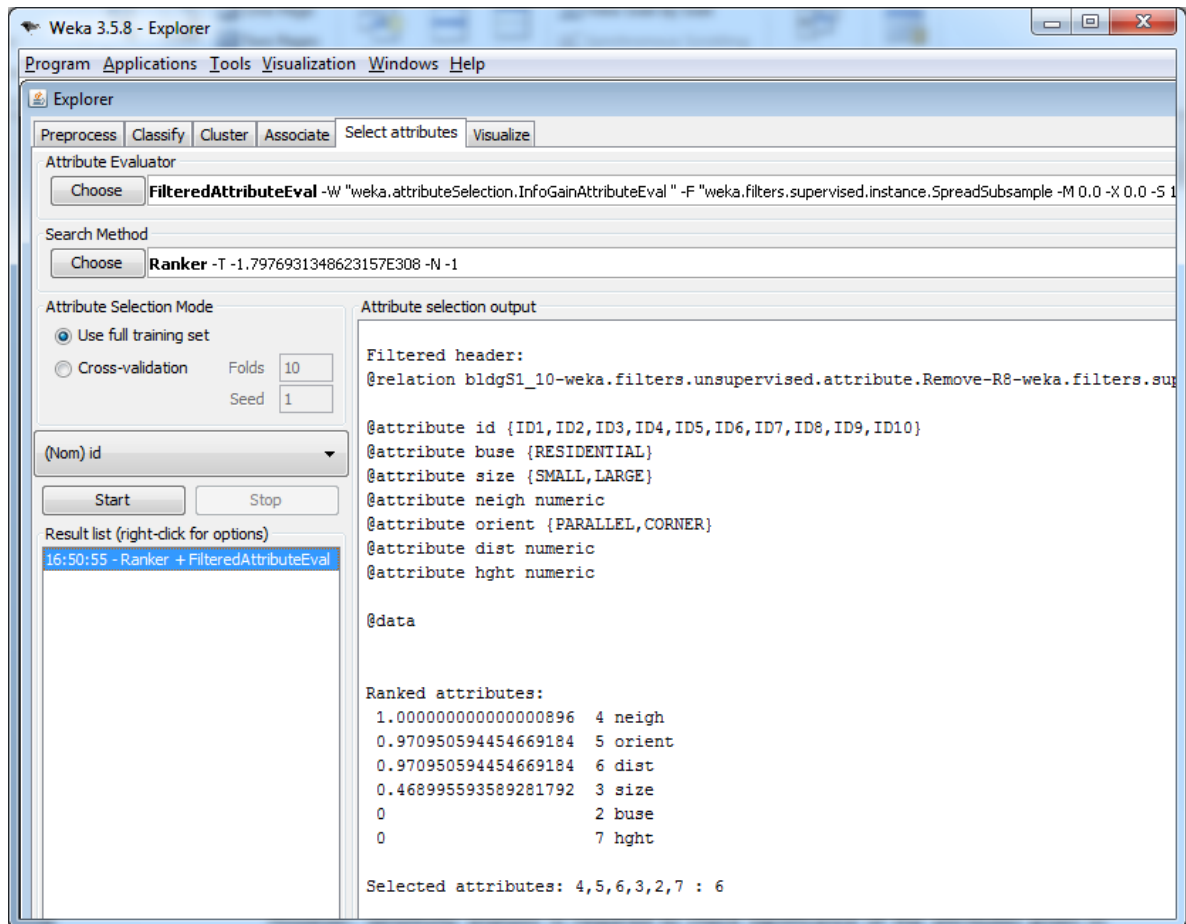


Figure F.2 Output of the sensitivity analysis in the WEKA GUI.

F.3 Attribute transformation in the WEKA software

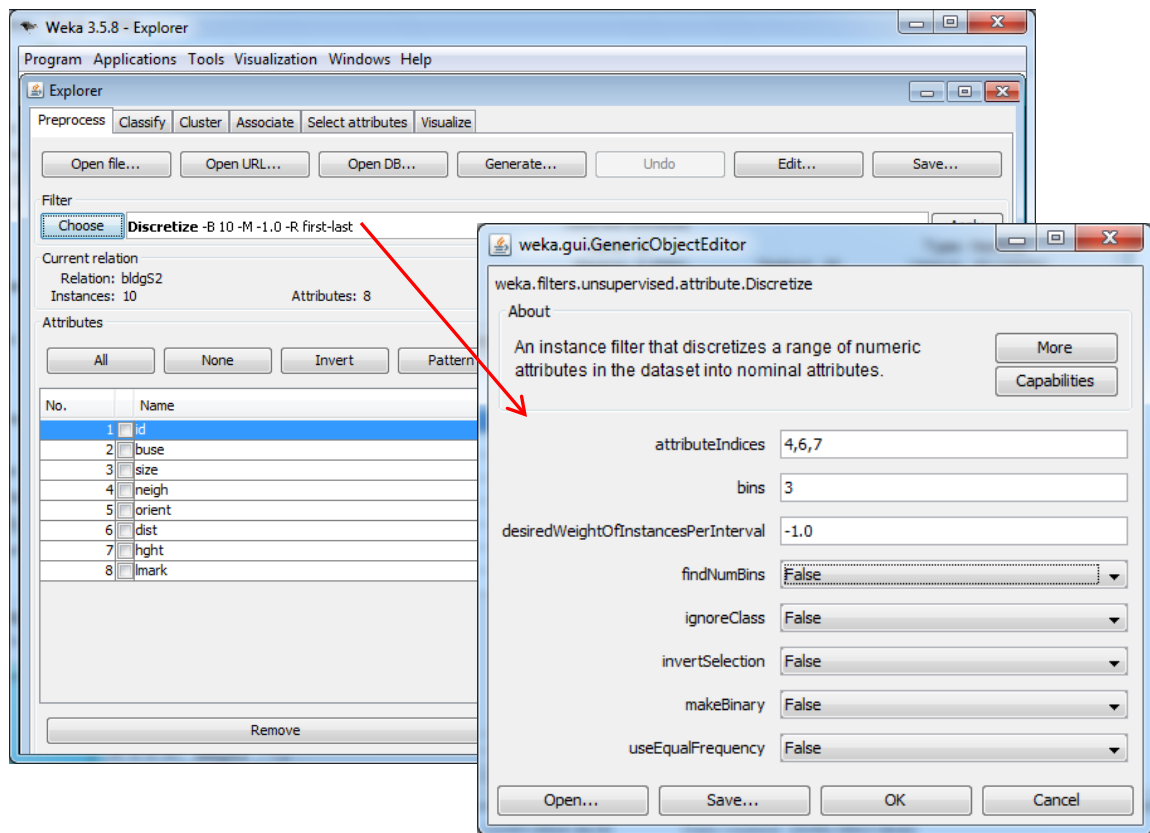


Figure F.3 Attribute transformation with the unsupervised discretization using the equal-width binning as shown in the input parameter menu.

F.4 Data mining user interface

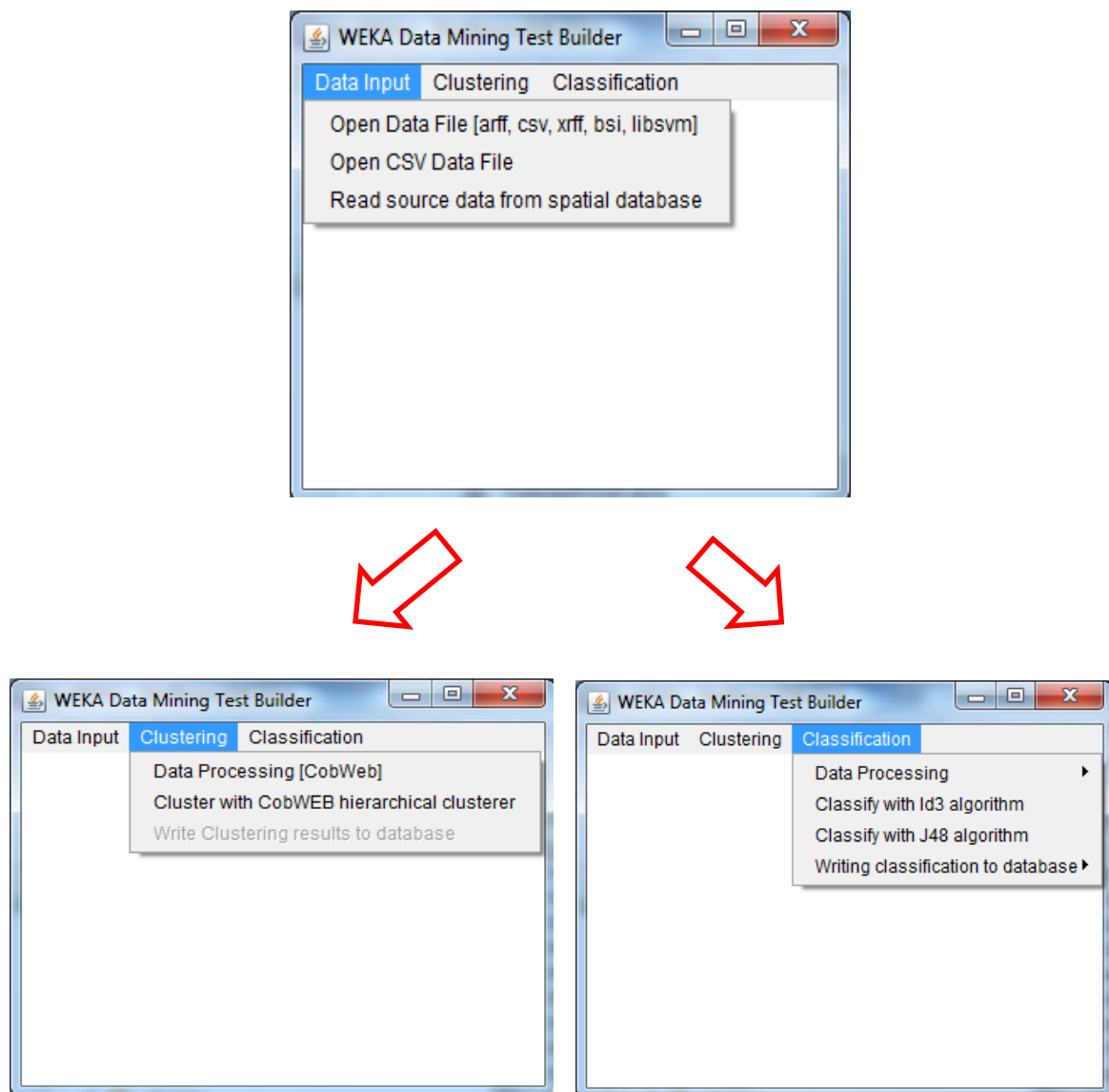


Figure F.4 Data mining UI for extracting the salient landmarks using the algorithms - CobWeb, ID3 and J48 - implemented with the open source WEKA Java APIs.

F.5 Enriched real test data (part of)

Table F.1 Enriched test data (part of) of a region surrounded by the road network. Data source: OS MasterMap.

class_id	size	dem	high	corners	diverse	side	orientation	elongation	orthogonal	mindst	rc	orientation	adj_neigh	neigh_density	mindst_n	av_or	ne	importance	mark
I1	69	10.5	8	0	335	0.9	1	9.8	angular	1		0.0042	0	0 ?	yes				
I2	47	9.8	4	0	336	0.5	1	12.4	across	1		0.0039	0	0 ?	no				
I3	47	10.8	6	0	335	0.5	1	11.0	angular	1		0.0045	0	0 ?	no				
I4	41	10.5	4	0	65	0.8	1	8.3	corner	1		0.002	0	1 ?	no				
I5	54	10.7	6	0	65	0.5	1	10.4	across	2		0.0036	0	0 ?	no				
I6	71	10.4	8	0	65	0.8	1	11.8	corner	2		0.0025	0	0 ?	no				
I7	47	10.7	4	0	336	0.5	1	13.0	angular	2		0.0038	0	0 ?	no				
I8	49	12.4	4	0	66	0.4	1	-1.0	none	1		0.0031	0	0 ?	no				
I9	70	10.3	8	0	335	0.9	1	12.3	across	1		0.0027	0	0 ?	no				
I10	69	10.7	8	0	335	0.9	1	12.3	across	1		0.0033	0	0 ?	no				
I11	52	10.8	6	0	335	0.5	1	10.7	across	2		0.0034	0	0 ?	no				
I12	51	10.8	6	0	334	0.5	1	9.3	across	1		0.0032	0	1 ?	no				
I13	53	10.5	6	0	335	0.5	1	11.0	across	2		0.0022	0	1 ?	no				
I14	51	10.8	6	0	335	0.5	1	11.0	across	2		0.0024	0	1 ?	no				
I15	53	12.4	4	0	66	0.4	1	-1.0	none	2		0.0023	0	1 ?	no				
I16	50	10.6	6	0	335	0.5	1	12.0	across	2		0.0046	0	0 ?	no				
I17	66	10.5	4	0	336	0.8	1	12.6	across	0		0.0038	3	1 ?	no				
I18	47	10.8	6	0	335	0.5	1	13.6	across	2		0.0046	0	0 ?	no				
I19	39	10.6	4	0	65	0.8	1	9.5	across	2		0.0022	0	1 ?	no				
I20	41	10.5	4	0	65	0.8	1	9.5	across	2		0.0023	0	1 ?	no				
I21	50	10.8	6	0	335	0.5	1	12.1	across	2		0.0046	0	0 ?	no				
I22	41	10.5	4	0	65	0.8	1	9.5	across	2		0.0024	0	1 ?	no				
I23	53	10.8	6	0	335	0.5	1	10.8	across	2		0.0034	0	0 ?	no				

F.6 User interface for the salient landmark evaluation

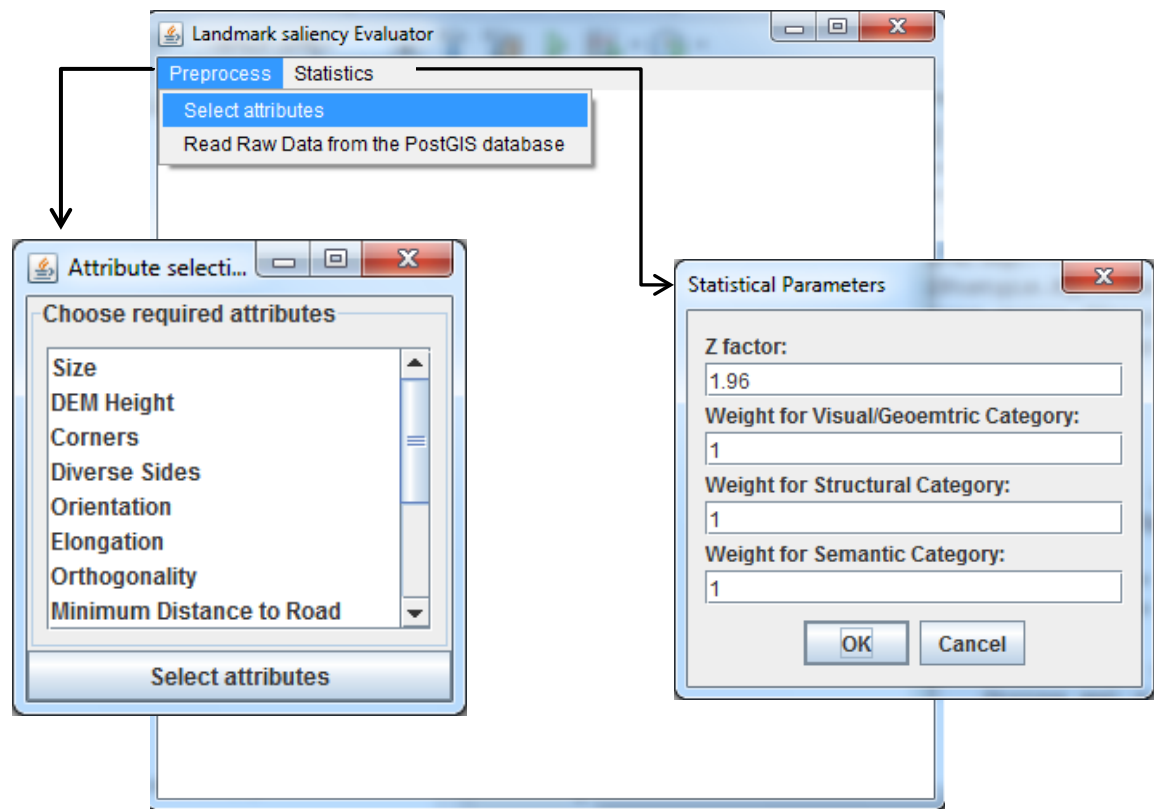


Figure F.5 User interface for the salient landmark evaluation based on the two frameworks of (a) Raubal and Winter (2002) and (b) Nothegger, Winter and Raubal (2004), and the method on the MAD developed in this research.

F.7 Results of the salient landmark evaluation at a decision point (Tower Hamlets area)

Table F.2 Results of the salient landmarks at a decision point.

idn	$\alpha 1$	$\alpha 2$	$\alpha 3$	$\alpha 4$	$\alpha 5$	$\alpha 6$	$\beta 1$	$\beta 2$	$\beta 3$	$\beta 4$	$\beta 5$	$\beta 6$	$\beta 7$	$\gamma 1$	$\sigma 1$	$\sigma 2$	$\sigma 3$	ϵ
1	12	82	8	3.33	0	0	0.098	2	344	0	1	0	312.5	0	0	0	0	0
2	21	40	4	2	0	0	0.105	2	68	0	0	0	181.82	0	0	0	0	0
3	23	62	6	2	0	0	0.128	1	68	0	2	0	196.08	0	0	0	0	0
4	23	61	6	2	0	0	0.13	2	68	0	0	0	188.68	0	0	0	0	0
5	29	404	8	2.5	0	0	0.083	1	72	1	0	0	322.58	1	1	1	1	1
6	31	175	4	1.25	0	0	0.125	2	342	1	1	0	277.78	0	1	1	0	0.67
7	26	335	4	3.33	0	0	0.167	1	72	1	3	0	256.41	0	1	1	0	0.67
8	34	570	24	1.11	0	0	0.057	4	72	1	1	0	303.03	1	2	1	1	1.33
9	31	603	14	1.43	0	0	0.125	3	340	1	1	0	357.14	0	2	1	0	1
10	31	270	12	2	0	0	0.061	2	342	1	1	0	312.5	1	1	1	1	1
11	21	50	4	1.67	0	0	0.161	1	68	0	0	0	238.1	0	0	0	0	0
12	24	101	6	3.33	0	0	0.238	1	338	0	2	0	256.41	0	0	1	0	0.33
13	21	99	5	2.5	0	0	0.238	2	342	1	3	0	256.41	0	0	2	0	0.67
14	25	77	6	3.33	0	0	0.092	3	343	0	0	0	344.83	0	0	0	0	0
15	13	54	6	5	0	0	0.147	1	342	1	1	0	243.9	0	0	1	0	0.33
16	13	65	4	3.33	0	0	0.11	2	342	1	1	0	256.41	0	0	1	0	0.33
17	13	46	8	5	0	0	0.132	2	342	1	1	0	256.41	0	0	1	0	0.33
18	13	54	4	2.5	0	0	0.1	2	344	0	1	0	303.03	0	0	0	0	0
19	13	68	4	2.5	0	0	0.106	3	343	1	0	0	357.14	0	0	1	0	0.33
20	13	207	4	2.5	0	0	0.167	1	73	1	1	0	270.27	0	1	1	0	0.67
21	23	290	17	2.5	0	0	0.118	5	343	1	0	0	357.14	1	2	1	1	1.33
22	25	47	8	2.5	0	0	0.213	1	73	1	0	0	370.37	1	0	1	1	0.67
23	17	6	4	5	0	0	0.125	0	72	1	0	3.2	322.58	0	0	2	0	0.67
24	21	49	10	1.25	0	0	0.116	3	343	0	0	0	322.58	0	0	0	0	0
25	13	76	4	3.33	0	0	0.208	2	343	0	0	0	344.83	0	0	0	0	0
26	29	67	6	1.67	0	0	0.066	2	342	1	1	0	312.5	0	0	1	0	0.33

α represents visual properties: $\alpha 1$ – DEM height, $\alpha 2$ – size, $\alpha 3$ – No. of corners, $\alpha 4$ – Inverse of elongation, $\alpha 5$ – orthogonal and $\alpha 6$ – diverse sides.

β represents structural properties: $\beta 1$ – Inverse of minimum distance to road, $\beta 2$ – No. of adjacent neighbours, $\beta 3$ – orientation to North, $\beta 4$ – Orientation to road, $\beta 5$ – Average orientation to neighbours, $\beta 6$ – Minimum distance to neighbour and $\beta 7$ – Inverse of neighbourhood density.

γ represents semantic properties: $\gamma 1$ – Importance.

$\sigma 1$ – Visual significance, $\sigma 2$ – Structural significance, $\sigma 3$ – Semantic significance, ϵ – Total significance.

G Pseudo codes of the developed algorithms

This section presents pseudo codes of the developed algorithms that contribute to generating focus maps.

G.1 Constrained algorithm on Delaunay triangulation

Input:

Building polygon geometry array: barray

Output:

Delaunay triangle array: delArray with enforcement of building edges as constraints

Main method: ConstrainedTriangulation(barray)

```
(1)   create a Point array: Point_dt[] //Point is the data type
(2)   int i = 0
(3)   for each building geometry:bgeom in barray {
(4)       Coordinate[] ca = bgeom.getCoordinates()
(5)       for (j = ca.getLength() - 2, j >= 0, j--) {
(6)           double X = ca[j].x
(7)           double Y = ca[j].y
(8)           arrpt[i] = new Point_dt[X, Y, 0.0]
(9)           sitepoints[i] = new Point_dt[X, Y, 0.0]
(10)      }
(11)  }
(12)  Geometry ch = calculateConvexHull(sitepoints) //creating convex hull of all building vertices
(13)  STRtree indx = buildSTRtreeIndex(bgeom) //creating an index of all building geometries
(14)  Triangle_dt[] triarray = DelaunayTriangulation(arrpt[i]) //invoking Delaunay triangulation library
(15)  for each triangle:triang in triarray {
(16)      boolean chk = checkPolygonIntersection(triang, indx) //check if edges intersect buildings
(17)      if (chk = false) then {
(18)          add triang into delArray
(19)      }
(20)  }
(21)  add buildings in barray and triangles in delArray into a single array:geomarray
(22)  union geometries in geomarray into a multi-geometry:mgeom
(23)  Geometry spcae = ch.getDifference(mgeom)//get the difference between convex hull and mgeom
(24)  if (space.getNumGeometries > 0) then {
(25)      for each polygon:poly in space
```

```

(26)         get poly.getCoordinates():pc
(27)         if (pc.getLength() >= 5) then { //polygon other than a triangle
(28)             for (j = pc.getLength() - 1, j >= 0, j--) {
(29)                 double X = ca[j].x
(30)                 double Y = ca[j].y
(31)                 arrpt[i] = new Point_dt[X, Y, 0.0]
(32)                 sitepoints[i] = new Point_dt[X, Y, 0.0]
(33)             }
(34)             //invoke Delaunay triangulation library
(35)             Triangle_dt[] triarray = DelaunayTriangulation(arrpt[i])
(36)             for each triangle:triang in triarray {
(37)                 boolean chk = checkPolygonIntersection(triang, indx)
(38)                 if (chk = false) then {
(39)                     add triang into delArray
(40)                 }
(41)             }
(42)         }
(43)     }
(44)     return delArray

```

Procedure **buildSTRtreeIndex**(Geometry geom)

```

(1)     STRtree bldindx = new STRtree() //instantiate bldindx
(2)     insert bounding box of geom into bldindx
(3)     return bldindx

```

Procedure **calculateConvexHull**(Array Coordinate[] ca)

```

(1)     create Geometry:geom from ca
(2)     ConvexHull ch = new ConvexHull(geom)//instantiate ch
(3)     Geometry gh = ch.getConvexHull() //creating convexhull geometry
(4)     return gh

```


Procedure **checkPolygonIntersection**(Triangle_dt triang, STRtree indx)

```
(1)    boolean av = false
(2)    get coordinates of triang into an ArrayList:alcoord
(3)    create triangle polygon geometry: trigeom
(4)    ArrayList indxlst = indx.query(trigeom.getEnvelopeinternal()) //retrieve close buildings
(5)    Iterator itl = indxlst.iterator()
(6)    while (itl.hasNext) {
(7)        Geometry bldg_poly = itl.Next()
(8)        if (trigeom.intersect(bldg_poly) = true then {
(9)            av = true
(10)           break
(11)        }
(12)    }
(13)    return av
```

G.2 Clustering algorithm

Input:

Building polygon geometry array: barray
Minimum distance threshold between buildings: vcdst
Medium distance threshold between buildings: mdst
Orientation difference threshold: odth
Similarity difference threshold: simdth
Target scale: double sf

Member:

Map med0
Map hmVCCluster
Map hmVFCluster
Map hmpMCluster

Output:

hmVCCluster with bid and cluster_id: VC-i / MS-i / MDS-I //re-assignment of values in a map
med0 with bid and label: "MD-0" / "ML-0" / "MDS-0" // re-assignment of values in a map
hmVFCluster with bid and label "VF-0" in a map

Main method: Clustering(barray)

```
(1)    ArrayList<String> gestalt = new ArrayList();//instantiate
(2)    Array Triarray = ConstrainedTriangulation(barray)
(3)    ArrayList<String> blnk = create adjacency relations between triangles //[bid_from, bid_to,dst]
(4)    for (i = 0, i < blnk.size(), i++) {
(5)        String lst = blnk.get(i)
(6)        split lst and get three values: bid1, bid2 and dst
(7)        get bgeom1and bgeom2 from bid1 and bid2
(8)        //get following Gestalt factors between two building geometries: bgeom1 and bgeom2
(9)        get orientation difference:ordiff using wall statistical algorithm by Duchene(2003)
(10)       get the similarity difference:simdifff using overlap ratio, compactness and Hausdorff
distance
(11)       // using target scale:sf
(12)       add Gestalt values [bid1,bid2,dst,ordiff,simdifff] into ArrayList: gestalt
(13)    }
(14)    ArrayList<String[ ]> mst = create Minimum Spannig Tree from gestalt
```

```

(15)   for (j = 0, j < mst.size(), j++) {
(16)       ArrayList<String> str = mst.get(i)
(17)       split str and get bid1, bid2 and dst
(18)       add bid1 and bid2 into String[] gf
(19)       ///creating Adjacency Matrix with proximity limits
(20)       String AdjMAT [ ] [ ] = createAdjacencyMatrix(vcdst, mdst, dst, gf)
(21)   }///end for

(22)   Map map0 = iterateFirstColumn (AdjMat[ ] [ ]) ///mapping 1st column ids
(23)   refineAdjMatColumn23 (AdjMat[ ] [ ], map0)
(24)   Map map1= iterateSecondColumn (AdjMat[ ] [ ]) ///mapping 2nd column ids
(25)   refineMapMed0 (map1,med0)
(26)   refineAdjMatColumn3 (AdjMat[ ] [ ], map1) ///This completes refining matrix for proximity
(27)   hmpVCCluster = ClusterColumnArray(0)
(28)   hmpMCluster = ClusterColumnArray(1)
(29)   hmpVFCluster = ClusterColumnArray(2)
(30)   ArrayList<ArrayList<String>> array_r = createDynamicMatrix (string AdjMat[ ] [ ])
(31)   ///using hmpMcluster
(32)   for each cluster in array_r {
(33)       enrich missing adjacency links using blnk array
(34)       ArrayList<String[ ]> mst1 = create Minimum Spannig Tree from cluster: weight: ordiff
(35)       run similar steps above from (15) to (28) based on orientation difference: small and large
(36)       ///using threshold: odth
(37)   }
(38)   create dynamicMatrix: ArrayList<ArrayList<String>> array_or for smaller orientations
(39)   for each cluster in array_or {
(40)       enrich missing adjacency links using blnk array
(41)       ArrayList<String[ ]> mst2 = create Minimum Spannig Tree from cluster:weight simdiff
(42)       run similar steps above from (15) to (28) based on the similarity difference: similar and
(43)       /// dissimilar using threshold: simdth
(44)   }

```

Procedure **createAdjacencyMatrix**(double vcdst, double mdst, double dst, String[] gf)

```
(1) List<String> alst = new ArrayList<String>() // instantiation
(2) String AdjMat [ ] [ ] = new String [alst.size()] [ ]
(3) String[] gfa = new String [3]
(4) if (dst <= vcdst) then {
(5)     gfa[0] = gf
(6)     gfa[1] = "0,0"
(7)     gfa[2] = "0,0"
(8) }
(9) else if (dst > vcdst) then {
(10)     gfa[0] = "0,0"
(11)     gfa[1] = gf
(12)     gfa[2] = "0,0"
(13) }
(14) else {
(15)     gfa[0] = "0,0"
(16)     gfa[1] = "0,0"
(17)     gfa[2] = gf
(18) }
(19) add gfa into alst
(20) AdjMat[ ] [ ] = alst.toArray(AdjMat[ ] [ ])
(21) return AdjMat[ ] [ ]
```

Procedure **iterateFirstColumn** (string AdjMat[] [])

```
(1) for (j = 0, j < AdjMat.length-1, j++) {
(2)     String fce = AdjMat [j] [0] //first column
(3)     split fce
(4)     String bid1 = val[0], String bid2 = val[1]
(5)     put (bid1,"1C") and (bid2,"1C") in Map:map0
(6) }
(7) return map0
```

Procedure void **refineAdjMatColumn23** (string AdjMat[] [], Map map0)

```
(1)    for (j = 0, j < AdjMat.length-1, j++) {
(2)        String ce3 = AdjMat [j] [2]
(3)        String[] gf3 = split ce3
(4)        String bid1_3 = gf3[0]
(5)        if (bid1_3 not equals ("0")) then {
(6)            String bid2_3 = gf3[1]
(7)            boolean b1_3 = map0.containsKey(bid1_3)
(8)            boolean b2_3 = map0.containsKey(bid2_3)
(9)            if (b1_3 = true and b2_3 = false) then {
(10)                AdjMat [j] [2] = bid2_3 + "," + "0"
(11)            }
(12)            if (b1_3 = false and b2_3 = true) then {
(13)                AdjMat [j] [2] = bid1_3 + "," + "0"
(14)            }
(15)            if (b1_3 = true and b2_3 = true) then {
(16)                AdjMat [j] [2] = "0" + "," + "0"
(17)            }
(18)        }
(19)        //now compare 2nd column with 1st column
(20)        String ce2 = AdjMat [j] [1]
(21)        String[] gf2 = split ce2
(22)        String bid1_2 = gf2[0]
(23)        if (bid1_2 not equals ("0")) then {
(24)            String bid2_2 = gf2[1]
(25)            boolean b1_2 = map0.containsKey(bid1_2)
(26)            boolean b2_2 = map0.containsKey(bid2_2)
(27)            if (b1_2 = true and b2_2 = false) then {
```

```

(28)             AdjMat [j] [1] = "0" + "," + "0"
(29)             put (b1_2,"0") in Map: med0
(30)         }
(31)         if (b1_2 = false and b2_2 = true) then {
(32)             AdjMat [j] [2] = bid1_2 + "," + "0"
(33)             put (b2_2,"0") in Map: med0
(34)         }
(35)         if (b1_2 = true and b2_2 = true) then {
(36)             AdjMat [j] [2] = "0" + "," + "0"
(37)         }
(38)     }
(39) } //end for

```

Procedure **iterateSecondColumn** (string AdjMat[] [])

```

(1)     for (j = 0, j < AdjMat.length-1, j++) {
(2)         String sce = AdjMat [j] [1] //second column
(3)         split sce
(4)         String bid1 = val[0], String bid2 = val[1]
(5)         put (bid1,"2C") and (bid2,"2C") in Map:map1
(6)     }
(7)     return map1

```

Procedure void **refineMapMed0** (Map map1, Map med0)

```
(1)   while (med0) {  
(2)       get mapkey: String med0.key  
(3)       boolean bidex = map1.containsKey(med0.key)  
(4)       if (bidex = true) then {  
(5)           remove med0.key and med0.value  
(6)       }  
(7)   }//end while
```

Procedure void **refineAdjMatColumn3** (string AdjMat[] [], Map map1)

```
(1)   for (j = 0, j < AdjMat.length-1, j++) {  
(2)       String ce3 = AdjMat [j] [2]  
(3)       String[] gf3 = split ce3  
(4)       String bid1_3 = gf3[0]  
(5)       if (bid1_3 not equals ("0") and bid2_3 not equals("0") then {  
(6)           String bid2_3 = gf3[1]  
(7)           boolean b1_3 = map1.containsKey(bid1_3)  
(8)           boolean b2_3 = map1.containsKey(bid2_3)  
(9)           if (b1_3 = true and b2_3 = false) then {  
(10)               boolean mz0 = med0.containsKey(bid2_3)  
(11)               if (mz0 = false) then {  
(12)                   AdjMat[j][2] = bid2_3 + "," + "0"  
(13)               }  
(14)               else  
(15)               {  
(16)                   AdjMat[j][2] = "0" + "," + "0"  
(17)               }  
(18)           }  
(19)       if (b1_3= false and b2_3= true) then {
```

```

(20)         boolean mz0 = med0.containsKey(bid1_3)
(21)         if (mz0 = false) then {
(22)             AdjMat[j][2] = bid1_3 + "," + "0"
(23)         }
(24)         else {
(25)             AdjMat[j][2] = "0" + "," + "0"
(26)         }
(27)     }
(28)     if (b1_3 = true and b2_3= true) then {
(29)         AdjMat[j][2] = "0" + "," + "0";
(30)     }
(31)     //This is to check both ids of column 3 non-available on column 2
(32)     //are available in hmpMzero// new
(33)     if (b1_3 = false and b2_3= false) then {
(34)         boolean bhmpMEx = med0.containsKey(b1_3)
(35)         boolean bhmpMEx1 = med0.containsKey(b2_3)
(36)         if (bhmpMEx = true and bhmpMEx1 == false) then {
(37)             AdjMat[j][2] = bid2_3 + "," + "0"
(38)         }
(39)         if (bhmpMEx1 = true and bhmpMEx = false) then {
(40)             AdjMat[j][2] = bid1_3 + "," + "0"
(41)         }
(42)         if (bhmpMEx = true and bhmpMEx1 = true) then {
(43)             AdjMat[j][2] = "0,0";
(44)         }
(45)     }
(46) }
(47) //This is to check links like e.g. "9,0". Check if the first id available in the second column.

```



```

(48)          //If available, the whole element is set to zero in the2nd column
(49)          else if (bid1_3 not equals("0") and bid2_3 equals("0")) then
(50)              boolean b1_3Exists = map1.containsKey(bid1_3)
(51)              if (b1_3Exists = true) then {
(52)                  AdjMat[j][2] = "0,0"
(53)              }
(54)          //This is to check if first id of column 3 exists in a single cluster values of the second
(55)              //column in med0 (all values checked)
(56)              boolean bhmpMEx = med0.containsKey(bid1_3)
(57)              if (bhmpMEx = true) then {
(58)                  AdjMat[j][2] = "0,0"
(59)              }
(60)          }
(61)      } //end for

```

//Grouping of buildings into clusters: Input Parameter is the column id which is either 0, 1 or 2;
//0 = First column, 1 = second column and 2 = third column by applying on refined AdjMat 2D Matrix.

procedure **ClusterColumnArray**(Integer ci)

```

(1)      Map hmapCluster = new HashMap() //instantiation
(2)      int kk = 1
(3)      int k = 0
(4)      for (j= 0, j <= AdjMat.length-1,j++) {
(5)          String fc = AdjMat [j] [ci]
(6)          String[] vc = split fc
(7)          String sbid1 = vc[0], String sbid2 = vc[1]
(8)          convert sbid1 and sbid2 into integers: ibid1and ibid2
(9)          if (ibid1 not equal 0 and ibid2 not equal 0) then {
(10)              boolean id1_bool = hmapCluster.containsKey(ibid1)

```

```

(11)         boolean id2_bool = hmapCluster.containsKey(ibid2)
(12)         if (id1_bool = false and id2_bool = false) then {
(13)             if (kk = 1) then {
(14)                 k++
(15)             }
(16)             put (ibid1, k) and (ibid2, k) in hmapCluster
(17)         }
(18)         else {
(19)             if (id1_bool = false) then {
(20)                 int kt = (Integer)hmapCluster.get(ibid2)
(21)                 put (ibid1, kt) in hmapCluster
(22)             }
(23)             if (id2_bool = false) then {
(24)                 int kt = (Integer)hmapCluster.get(ibid1)
(25)                 put (ibid2, kt) in hmapCluster
(26)             }
(27)         }
(28)     } else if (ibid1 not equal 0 and ibid2 equals 0) then {
(29)         put (ibid1, 0) in hmapCluster
(30)     }
(31)     else //denotes 0,0 {
(32)         kk = 1
(33)     }
(34) } //end for
(35) return hmapCluster

```

```

Procedure createDynamicMatrix (string AdjMat[ ] [ ]) //2D dynamic matrix
(1)      ArrayList<ArrayList<String>> array_r = new ArrayList<ArrayList<String>>()
(2)      get the number of medium clusters: nc from map: hmpMCluster
(3)      for (i = 1, i <= nc, ++i) {
(4)          ArrayList<String> array_c = new ArrayList<String>()
(5)          for(j=0, j<=AdjMat.length-1, j++) {
(6)              String sc = AdjMat[j][1] //second column
(7)              String[ ] vc = split sc
(8)              get int bid1 from vc[0]
(9)              while (hmpMCluster) { //iterate
(10)                  get key:v1 //building_id
(11)                  get value:v2 //cluster_id
(12)                  if (v2 = i and ibid1 =v1) then {
(13)                      add sc into array_c
(14)                  }
(15)              } //end while
(16)          } //end for
(17)          add array_c into array_r
(18)      } //end for
(19)      return array_r //2D array

```

G.3 Cluster shape enrichment

Input:

Building cluster multipolygon geometry: cgeom

Orientation threshold: orthold

Angle precision: angpr

Output:

boolean chkortho {false – non-orthogonal, true – orthogonal}

Main method: Clustershape (Geometry cgeom, double orthold, double angpr)

```
(1)  for each cluster:cgeom in PostGres database {
(2)      select cluster:cgeom where cluster_label = 'VC' or 'MS' or 'MDS'
(3)      ArrayList<Geometry> algeom = EnrichCluster(cgeom)
(4)      boolean chkortn = CheckOrientation(cgeom, angpr, ortn)
(5)      if (chkortn = true) then {
(6)          boolean chkortho = CheckOrthogonality(cgeom, angpr)
(7)      }
(8)      else {
(9)          chkortho = false
(10)     }
(11) }
(12) return chkortho
```

Procedure **EnrichCluster**(Geometry mgeom)

```
(1)  explode mutipolygon geometry: mgeom into polygon geometry array:ArrayList<algeom>
(2)  ArrayList<Geometry> alg = new ArrayList<Geometry>() //instantiate ArrayList
(3)  create concave hull: geometry chull from algeom with Java library
(4)  for (i = 0, i < algeom.size(), i++) {
(5)      Geoemtry bldg = algeom.get(i)
(6)      if (bldg touches chull by a point or line) then {
(7)          add bldg into alg
(8)      }
(9)  }
(10) return alg
```

Procedure **CheckOrientation** (ArrayList<Geometry> ageom, double angpr, double orthold)

```
(1)    boolean smalldiff = true
(2)    int countover45 = 0, countless45 = 0
(3)    List<Double> lstorn = new List<Double>() //object instantiation
(4)    for (i = 0, i < ageom.size(), i++) {
(5)        Geoemtry geom = ageom.get(i)
(6)        double bortn = get orientation on wall statistical algorithm by Duchene (2003)
(7)        if (bortn > 45) then {
(8)            countover45++
(8)        }
(9)        else {
(10)            countless45++
(11)        }
(12)        add bortn into lstorn
(13)    }
(14)    lstorn = sort lstorn in ascending order
(15)    get the difference in orientation: diffort using min and max values from lstorn
(16)    if (diffort > 45) then {
(17)        List<Double> lstnew = new ArrayList<Double>()
(18)        for (i = 0, i < lstorn.size(), i++) {
(19)            double tval = lstorn.get(i)
(20)            if (cntover45 >= cntless45) then {
(21)                if (tval < 45) then {
(22)                    tval = tval + 90
(23)                }
(24)                else {
(25)                    add tval into lstnew
(26)                }
(27)            else {
(28)                if (tval > 45) then {
(29)                    tval = 90 – tval
(30)                }
(31)                else {
(32)                    add tval into lstnew
(33)                }
(34)            }
(35)    sort lstnew in ascending order
```

```

(36)  get the difference in orientation: diffort using min and max values from lstnew
(37)  }
(38)  if (diffort > orthold) then {
(39)      smalldiff = false
(40)  }
(41)  return smalldiff

```

Procedure **CheckOrthogonality** (ArrayList<Geometry> ageom, double angpr)

```

(1)    boolean orthocheck = false
(2)    if (ageom.isEmpty() = false) then {
(3)        for (i = 0, i < ageom.size(), i++) {
(4)            get double:confidence_indct from wall statistical algorithm by Duchene (2003)
(5)            if (confidence_indct < 80) then {
(6)                orthocheck = false
(7)                break
(8)            }
(9)        }
(10)   }
(11)   return orthocheck

```

G.4 Symbolization algorithm

Input:

Building cluster geometry: bclust

Minimum side length: mlength

Output:

Symbolized polygon: spoly

Main method: symbolization(bclust)

- (1) **create** concave hull:chull on building cluster:bclust
- (2) **for each** building edge **in** bclust {
- (3) **find** the edge:bedge that touches chull by a line
- (4) **add** bedge into ArrayList<edge>:edges
- (5) }
- (6) geom_{x-y}, BB_{x-y}, rotang = **create_geom_and_bounding_box_X-Y**(edges, bclust)
- (7) **if** (both length and width of BB_{x-y} < mlength) **then** {
- (8) **create** a square polygon:sqpoly with a minimum length:mlength
- (9) **rotate** sqpoly around its centroid with -rotang back: spoly
- (10) }
- (11) **if** (length of BB_{x-y} < mlength) **then** {
- (12) Geometry poly = **enlargealongY**(BB_{x-y}, mlength)
- (13) **rotate** poly around its centroid with -rotang back: spoly
- (14) }
- (15) **if** (width of BB_{x-y} < mlength) **then** {
- (16) Geometry poly = **enlargealongX**(BB_{x-y}, mlength)
- (17) **rotate** poly around its centroid with -rotang back: spoly
- (18) }
- (19) **return** spoly

Procedure **create_geom_and_bounding_box_X-Y**(Arraylist<edge> edges, geometry bclust)

- (1) Object obj[] = new Object[3]//creating an array of objects
- (2) **calculate** centroid:gcn of cluster geometry:bclust
- (3) **calculate** maximum local weighted orientation of edges:rotang
- (4) **rotate** bclust around point:gcn by rotang to orient cluster in X-Y direction
- (5) calculate minimum bounding box of the oriented geom in X-Y: BB_{X-Y}
- (6) obj[0] = geom_{X-Y}
- (7) obj[1] = BB_{X-Y}
- (8) obj[2] = rotang
- (9) **return** obj //return multiple values: geom_{X-Y}, BB_{X-Y} , rotang

Procedure **enlargealongY**(Geometry BB_{X-Y} , double dist)

- (1) **get** min and max coordinates of BB_{X-Y}
- (2) **create** an adjoining polygon:tpoly on top of BB_{X-Y} with min and max coordinates and $0.5 * dist$
- (3) **create** an adjoining polygon:bpoly at bottom of BB_{X-Y} with min and max coordinates and $0.5 * dist$
- (4) **union** BB_{X-Y} , tpoly and bpoly to create a polygon symbol:sym
- (5) **return** sym

Procedure **enlargealongX**(Geometry BB_{X-Y} , double dist)

- (1) **get** min and max coordinates of BB_{X-Y}
- (2) **create** an adjoining polygon:rpoly to the right of BB_{X-Y} with min and max coordinates and $0.5 * dist$
- (3) **create** an adjoining polygon:lpoly to the left of BB_{X-Y} with min and max coordinates and $0.5 * dist$
- (4) **union** BB_{X-Y} , rpoly and lpoly to create a polygon symbol:sym
- (5) **return** sym

G.5 Squaring algorithm

Input:

Building amalgam polygon: geom

Output:

Squared building amalgam polygon: sqgeom

Main method: squaring(geom)

```
(1)   geomX-Y, BBx, rotang. = create_geom_and_bounding_box_X-Y(geom)
(2)   calculate orientations along X and Y in BBX-Y
(3)   coordinate array:carray = geomX-Y.getcoordinate()
(4)   for (i = 0; i < carray.length-1, i++) {
(5)       //starting and end point coordinates of an edge
(6)       p1_x = getcoordinate.carray[i].x
(7)       p1_Y = getcoordinate.carray[i].y
(8)       p2_x = getcoordinate.carray[i+1].x
(9)       p2_Y = getcoordinate.carray[i+1].y
(10)      calculate edge distance:edst = len(p1,p2)
(11)      create four line strings each along X and Y directions from points p1 and p2 with distance: edst
(12)      add line strings at point p1 into an array:ap1
(13)      add line strings at point p2 into an array:ap2
(14)      coordinate array:carray = get_two_new_points(linestring array:lisa1, linestyle array:lisa2)
(15)      Point: np1 = create point carray[0]
(16)      Point:np2 = create point carray[1]
(17)      if (p1 != NULL and p2 != NULL) then {
(18)          create three line strings joining p1, p2 and np1
(19)          polygonize line strings to create ear-polygon:ep1 with points p1, p2, np1
(20)          create three line strings joining p1, p2 and np2
(21)          polygonize line strings to create ear-polygon:ep2 with points p1, p2, np2
(22)          find ear-polygon: ep that is outside geomX-Y out of ep1 and ep2
(23)          add ep into an geometry array:ageom }
(24)  }//end for
(25)  add geomX-Y into ageom
(26)  merge ear-polygons in ageom:mgeom
(27)  rotate mgeom with -rotang around its centroid back:sqgeom
(28)  return sqgeom
```

Procedure **create_geom_and_bounding_box_X-Y**(geom)

- (1) Object obj[] = new Object[3]//creating an array of objects
- (2) **calculate** centroid:gcn of polygon geometry:geom
- (3) **calculate** maximum local weighted orientation:rotang
- (4) **rotate** geom around point:gcn by rotang to orient in X-Y direction
- (5) calculate minimum bounding box of the oriented geom in X-Y: BB_{X-Y}
- (6) obj[0] = geom_{X-Y}
- (7) obj[1] = BB_{X-Y}
- (8) obj[2] = rotang
- (9) **return** obj //return multiple values: geom_{X-Y}, BB_{X-Y}, rotang

Procedure **get_two_new_points**(linestring array:lsa1, linestyle array:lsa2)

- (1) **for each** linestyle:ls1 in lsa1 {
- (2) **for each** linestyle:ls2 in lsa2 {
- (3) **if** ls1.intersects(ls2) **then** {
- (4) **get** coordinate of intersecting point:c1
- (5) **add** c1 into a coordinate array:array
- (6) }
- (7) }
- (8) }
- (9) **for each** linestyle:ls2 in lsa2 {
- (10) **for each** linestyle:ls1 in lsa1 {
- (11) **if** ls2.intersects(ls1) **then** {
- (12) **get** coordinate of intersecting point:c2
- (13) **add** c2 into a coordinate array:array
- (14) }
- (15) }
- (16) }
- (17) **return** array

G.6 Enlargement algorithm

Input:

Squared building amalgam polygon: sqgeom

Filling width: fw

Minimum width of narrow sections: minw

Output:

Squared and enlarged building amalgam polygon: famalgam

Main method: enlargement(sqgeom, fw)

- (1) Object[] obj = **create_geom_and_bounding_box_X-Y**(sqgeom)
- (2) Geometry geom_{x-y} = obj[0], Geoemtry BB_{x-y} = obj[1], double rotang = obj[2]
- (3) Geometry array: ipgeom = **create_inner_poly**(geom_{x-y})
- (4) Sorted array<Double> xlst = **sort_centroid_X_strips**(ipgeom)
- (5) Sorted array<Double> ylst = **sort_centroid_Y_strips**(ipgeom)
- (6) ArrayList<ArrayList<Geometry>> X_stack = **store_X_strip_stack** (ipgeom, ylst, minw)
- (7) ArrayList<ArrayList<Geometry>> Y_stack = **store_Y_strip_stack** (ipgeom, xlst, minw)
- (8) ArrayList<Geometry> opgeom = **create_outer_poly**(geom_{x-y}, BB_{x-y})
- (9) ArrayList<ArrayList<Geometry>> fillX = **getFillingStack_X**(X_stack, opgeom, minw)
- (10) ArrayList<ArrayList<Geometry>> fillY = **getFillingStack_Y**(Y_stack, opgeom, minw)
- (11) **union** all filling slices in fillX and fillY ArrayLists together: ufslices
- (12) **union** ufslices with geom_{x-y}: amal
- (13) **rotate** amal with -rotang back around its centroid: famalgam
- (14) **return** famalgam

Procedure **create_geom_and_bounding_box_X-Y**(geom)

- (1) Object obj[] = new Object[3]//creating an array of objects
- (2) **calculate** centrod:gcen of polygon geometry:geom
- (3) **caluculate** maximum local weighted orientation:rotang
- (4) **rotate** geom around point:gcen by rotang to orient in X-Y direction
- (5) **calculate** minimum bounding box of the oriented geom in X-Y: BB_{x-y}
- (6) obj[0] = geom_{x-y}
- (7) obj[1] = BB_{x-y}
- (8) obj[2] = rotang
- (9) **return** obj //return multiple values: geom_{x-y}, BB_{x-y}, rotang

Procedure **create_inner_poly**(geom_{x-y})

- (1) **for each** edge:bedge **in** geom_{x-y} {
- (2) **extend** bedge from both ends to intersect polygon:geom_{x-y}
- (3) }
- (4) **polygonize** all extended line strings to create polygon strips:ips inside geom_{x-y}
- (5) **add** polygon strips:ips into a geometry array:ipgeoma
- (6) **return** ipgeoma

Procedure **sort_centroid_X_strips**(geometry array:ipgeom)

- (1) **create** an arraylist<Double>:xlst
- (2) **for** (i=0, i < ipgeom.length, i++) {
- (3) Geometry ig = ipgeom.get(i)
- (4) X = ig.getCentroid.x
- (5) add X into array:xlst
- (6) }
- (7) **sort** xlst in ascending order
- (8) **return** xlst

Procedure **sort_centroid_Y_strips**(geometry array:ipgeom)

- (1) **create** an arraylist<Double>:ylst
- (2) **for** (i=0, i < ipgeom.length, i++) {
- (3) Geometry ig = ipgeom.get(i)
- (4) Y = ig.getCentroid.y
- (5) **add** Y into array:ylst
- (6) }
- (7) **sort** ylst in ascending order
- (8) **return** ylst

Procedure **store_X_strip_stack** (Arraylist<Geometrey> ipgeom, Arraylist<Double> xlst, double minw)

```
(1)   create Arraylist<Arraylist<Geometry>> x_stack_geom
(2)   for (i=0, i < ylst.size, i++) {
(3)       create Arraylist<Geometry> algeom
(4)       Cen_Y = ylst.get(i)
(5)       for (j=0, j < ipgeom.length, j++) {
(6)           Y_gcen = ipgeom.getCentrid.Y
(7)           width = ipgeom.getwidth
(8)           if (Cen_Y = Y_gcen and width < minw) then {
(9)               algeom.add(ipgeom)
(10)          }
(11)      }
(12)      if (algeom.size > 1) then {
(13)          x_stack_geom.add(algeom)
(14)      }
(15)  }
(16)  return x_stack_geom
```

Procedure **store_Y_strip_stack** (Arraylist<Geometrey> ipgeom, Arraylist<Double> ylst, double minw)

```
(1)   create Arraylist<Arraylist<Geometry>> y_stack_geom
(2)   for (i=0, i < xlst.size, i++) {
(3)       create Arraylist<Geometry> algeom
(4)       Cen_X = xlst.get(i)
(5)       for (j=0, j < ipgeom.length, j++) {
(6)           X_gcen = ipgeom.getCentrid.X
(7)           width = ipgeom.getwidth
(8)           if (Cen_X = X_gcen and width < minw) then {
(9)               algeom.add(ipgeom)
(10)          }
(11)      }
(12)      if (algeom.size > 1) then {
(13)          Y_stack_geom.add(algeom)
(14)      }
(15)  }
(16)  return Y_stack_geom
```

Procedure **getFillingStack_X**(Arraylist<Arraylist<Geometry>>>X_stack_geom, Arraylist<Geometry> opgeom, double minw)

```

(1)   Arraylist<Arraylist<Geometry>>> fillstack_X
(2)   for each strip in X_stack_geom {
(3)       Arraylist<Geometry> fillslice
(4)       for each slice in strip {
(5)           for each outerstrip in opgeom {
(6)               if (two outstrips exist on top and bottom on slice) then {
(7)                   union slices in strip:ustrip
(8)   create two strips on either side of the ustrip //to satisfy
(9)   //minimum filling width
(10)                  add strip_t into fillslice
(11)                  add strip_b into fillslice
(12)                  break
(13)              }
(14)              if (only one outstrip exist on top of slice) then {
(15)                  union slices in strip:ustrip
(16)                  if (a point just outside bottom of slice is out of  $BB_{x-y}$  = true) then {
(17)                      create a strip on top of the slice //to satisfy minimum
(18)                      // filling width
(19)                      add strip_b into fillslice
(20)                  }
(21)              }
(22)              if (only one outstrip exists on bottom of slice) then {
(23)                  union slices in strip:ustrip
(24)                  if (a point just outside right of slice is out of  $BB_{x-y}$  = true) then {
(25)                      create a strip on top of slice //to satisfy minimum filling
(26)                      //width
(27)                      add strip_t into fillslice
(28)                  }
(29)              }
(30)          }
(31)      }
(32)  }
(33)  add fillslice into fillstack_X //2D Arraylist
(34)  return fillstack_X

```

Procedure **getFillingStack_Y**(Arraylist<Arraylist<Geometry>>>Y_stack_geom, Arraylist<Geometry> opgeom, double minw)

```

(1)   Arraylist<Arraylist<Geometry>>> fillstack_Y
(2)   for each strip in Y_stack_geom {
(3)       Arraylist<Geometry> fillslice
(4)       for each slice in strip {
(5)           for each outerstrip in opgeom {
(6)               if (two outstrips exist to left and right of slice) then {
(7)                   union slices in strip:ustrip
(8)   create two strips on either side of the ustrip //to satisfy
(9)   //minimum filling width
(10)                  add strip_l into fillslice
(11)                  add strip_r into fillslice
(12)                  break
(13)              }
(14)              if (only one outstrip exist on left of slice) then {
(15)                  union slices in strip:ustrip
(16)                  if (a point just outside right of slice is out of  $BB_{x,y}$  = true) then {
(17)                      create a strip on left of slice //to satisfy minimum
(18)                      // filling width
(19)                      add strip_l into fillslice
(20)                  }
(21)              }
(22)              if (only one outstrip exists on right of slice) then {
(23)                  union slices in strip:ustrip
(24)                  if (a point just outside left of slice is out of  $BB_{x,y}$  = true) then {
(25)                      create a strip on right of slice //to satisfy minimum
(26)                      //filling width
(27)                      add strip_r into fillslice
(28)                  }
(29)              }
(30)          }
(31)      }
(32)  }
(33)  add fillslice into fillstack_X //2D Arraylist
(34)  return fillstack_X

```

G.7 Simplification algorithm

Input:

Building geometry polygon: geom

Simplification edge distance threshold: bethv

Output:

Simplified building geometry polygon: sgeom

Main method: simplification(geom, bethv)

```
(1)   Object[] obj = create_geom_and_bounding_box_X-Y(geom)
(2)   Geometry geomx-y = obj[0], Geometry BBx-y = obj[1], double rotang = obj[2]
(3)   boolean running = true
(4)   while(running) {
(5)       Map<Vertex,Coordinate> pmap = concave_identifier(geomx-y)
(6)       Geometry array:Cp = create_outer_poly(geomx-y, BBx-y)
(7)       ArrayList<Geometry> ageom = polygons_to_merge(pmap, Cp, bethv)
(8)       if (ageom is not empty) then {
(9)           ugeom = union array of polygons:ageom with initial polygon:geomx-y
(10)      }
(11)      else {
(12)          running = false
(13)      }
(14)  }//end while
(15)  rotate ugeom by value: -rotang around its centroid back to the original orientation to get sgeom
(16)  return sgeom
```

Procedure **create_geom_and_bounding_box_X-Y**(geom)

```
(1)   Object obj[] = new Object[3]//creating an array of objects
(2)   calculate centroid:gcen of polygon geometry:geom
(3)   calculate maximum local weighted orientation:rotang
(4)   rotate geom around point:gcen by rotang to orient in X-Y direction
(5)   calculate minimum bounding box of the oriented geom in X-Y: BBx-y
(6)   obj[0] = geomx-y
(7)   obj[1] = BBx-y
(8)   obj[2] = rotang
(9)   return obj //return multiple values: geomx-y, BBx-y, rotang
```


Procedure **concave_identifier**(geom_{x-y})

```

(1)   Array<Coordinates> ac = get coordinates.geomx-y
(2)   temp_angdiff = -1; vid = 0//vertex id
(3)   for each coordinate in ac {
(4)       get angle difference:angdiff between the two edges connecting the coordinate
(5)       if angdiff = 900 and temp_angdiff != 900 then {
(6)           map.put(k,coordinate)//concave vertex_id and its coordinate
(7)           temp_angdiff = angdiff
(8)       }
(9)   }
(10)  return map

```

Procedure **create_outer_poly**(geom_{x-y},BB_{x-y})

```

(1)   get the difference between source geometry:geomx-y and its BBx-y to output polygon:dp
(2)   create inner line strings of dp by extending its each edge to meet opposite edge if intersected
(3)   polygonize line strings to form inner polygons:ips
(4)   add ips into a geometry array: ipgeom
(5)   return ipgeom

```

Procedure **polygons_to_merge**(Map<Vertex,Coordinate> pmap, geometry array C_p, bethv)

```

(1)   iterator it pmap
(2)   while (it.hasNext()) {
(3)       get coordinate:value
(4)       for each geometry in Cp
(5)           if (value is on cp) then {
(6)               get envelope: envp
(7)               if (envp.length <= bethv or envp.width <=bethv) then
(8)                   {
(9)                       add envp into a geometry array: aenvp
(10)                  }
(11)          }
(12)      }
(13)  }//end while
(14)  return aenvp

```

G.8 Building aggregation with orthogonal sides

Input:

Building cluster (orthogonal) multipolygon geometry: cgeom

Minimum distance threshold between buildings: mindst

Minimum side length of a building: minlength

Filling threshold: fth

Simplification threshold: simth

Output:

Building amalgam array: ArrayList<Geometry> bamal

Main method: OrthogonalAmalgam(bclust, mindst, minlength, fth, simth)

```
(1)  ArrayList<String> blnk = new ArrayList<String>()
(2)  ArrayList<Geometry> triarray = new ArrayList<Geometry>()
(3)  ArrayList<Geometry>:bamal
(4)  for each cluster:cgeom in PostGIS database {
(5)      Geometry symbol = symbolization(cgeom)
(6)      if (symbol = null) then {
(7)          Geometry buffout = buffer_operation(cgeom,mindst,-mindst)
(8)          if (buffout.GeometryType = "Polygon") then {
(9)              Geometry sqgeom = squaring(buffout)
(10)             Geometry engeom = enlargement(sqgeom,fth)
(11)             Geometry simpg = simplification(engeom,simth)
(12)             add simpg into ArrayList<Geometry>:bamal
(13)         }
(14)     else if (buffout.GeometryType = "Multipolygon") then {
(15)         explode buffout into separate polygon array:ArrayList<Geometry> alg
(16)         Object[] val4 = recreatingGeometry(alg)
(17)         Map idgeom = val4[0]
(18)         Map geomid = val4[1]
(19)         Map coordid = val4[2]
(20)         Map STRindx = val4[3]
(21)         clear blnk //empty building link array
(22)         clear triarray //empty building link array
(23)         ArrayList<Geometry> triarray = ConstrainedTriangulation(alg)
(24)         get adjacency relations between each pair of buildings in blnk array
```

```

(25)         remove duplicate links in blnk
(26)     ArrayList<Geometry> spacetri = getBridges4Squaring(blink, coordid,
(27)         STRindx, geomid, idgeom, mindst)
(28)     ArrayList<Geometry> candidate_tris = getCandidateTriangles(spacetri,
(29)         mindst)
(30)     ArrayList<ArrayList<Geometry>> brdg_clusters =
(31)         getBridgingClusters(candidate_tris)
(32)     ArrayList<ArrayList<Geometry>> filter_brdg =
(33)         filterBridges(brdg_clusters)
(34)     convert filter_brdg array into a single ArrayList<Geometry>: tribrdgs
(35)     Geometry uamal = union tribrdgs
(36)     if (uamal = "Multipolygon") then {
(37)         Geometry buffg = bufferplus(uamal, +0.1)
(38)         get outergeom: outgeom of buffg
(39)         Geometry sqgeom = squaring(outgeom)
(40)         double fthv = fth + 2 * (0.1)
(41)         Geometry enlarge = enlargement(sqgeom,fthv)
(42)         Geometry refenlg = bufferplus(uamal,-0.1)
(43)         remove inter vertices if any between edges of refenlg
(44)         Geometry simpkg = simplification(refenlg,simth)
(45)         add simpkg into ArrayList<Geometry>:bamal
(46)     } else {
(47)         extract outer plygon:outpoly of simpkg
(48)         Geometry sqgeom = squaring(outpoly)
(49)         Geometry enlg = enlargement(sqgeom,fth)
(50)         Geometry simpkg = simplification(enlg)
(51)         add simpkg into ArrayList<Geometry>:bamal
(52)     }
(53) }
(54) }
(55) else {
(56)     add Geometry:symbol into ArrayList<Geometry>:bamal
(57) }
(58) }
(59) return bamal

```

Procedure **buffer_operation**(Geometry geom, double dst1, double dst2)

```
(1)    BufferParameter bp = new BufferParameter()//instantiate buffer parameter:bp
(2)    set bp.EndCapStyle(cap_square)
(3)    set bp.JoinStyle(flat_edge_corners)
(4)    BufferBuilder buffb = new BufferBuilder(bp)// instantiate buffer builder:buffb
(5)    Geometry buffgeom = buffb.buffer (geom, dst1)//positive buffer
(6)    Geometry fbp = buffb.buffer (buffgeom, dst2)//dst2 = -dst1 (negative buffer)
(7)    return fbp //return buffered amalgam
```

Procedure **recreatingGeometry**(ArrayList<Geometry> ageom)

```
(1)    //object instantiation
(2)    Object obj = new Object[4]
(3)    Map<String,Geometry> idgeom = new HashMap<String,Geometry>()
(4)    Map<Geometry,String> geomid = new HashMap<Geometry,String>()
(5)    Map<Coordinate, List<String>> cid = new HashMap<Coordinate, List<String>>()
(6)    STRtree strIndx = new STRtree()
(7)    //iterating geometries
(8)    for (i=0, i < ageom.size(), i++) {
(9)        put poygon_id: (i+1) and polygon geometry:ageom.get(i) into map:idgeom
(10)       put polygon geometry:ageom.get(i) and poygon_id: (i+1) into map:geomid
(11)       get outer polygon:geom_outer
(12)       insert geom_outer into strIndx
(13)       Coordinate[] gcoords = geom_outer.getCoordinates()
(14)       for (j = gcoords.length-2, j >= 0, j--) {
(15)           if (cid = null) then {
(16)               add Stirng(i+1) into new List:nl
(17)               put gcoords[j] and nl into map: cid
(18)           } else
(19)           {
(20)               get String(i+1) from map: cid from gcoords[j]
(21)               add Stirng(i+1) into new List:nl
(22)           }
(23)       }//end for
(24) }//end for
(25) return obj
```

Procedure **getBridges4Squaring**(ArrayList<String> blink, Map<Coordinate, List<String> clst, STRtree indx, Map<Geometry,String> geomid, Map<String,Geometry> idgeom, double mindst)

```

(1)    ArrayList<Geometry> spaceTri = new ArrayList<Geometry>()//instantiate array geometry
(2)    for (i=0, i < blink.size(), i++) {
(3)        get bid1, bid2 and distance:dst between bid1 and bid2 from blink array
(4)        Geometry bldg1 = idgeom.get(bid1)
(5)        Geometry bldg2 = idgeom.get(bid2)
(6)        Geometry ugeom = union bldg1 and bldg2
(7)        if (dst <= mindst) then {
(8)            ArrayList<Geometry> alg = run Conforming Delaunay triangulation on ugeom
(9)            for each triangle: tri in alg {
(10)                if (tri_ver1 != tri_ver2) then {
(11)                    add triangle:tri into spaceTri
(12)                }
(13)            }
(14)        }
(15)    }
(16)    return spaceTri

```

Procedure **getCandidateTriangles**(ArrayList<Geometry> spacegeom, double mindst)

```

(1)    ArrayList<Geometry> can_tri = new ArrayList<Geometry>()//instantiate geometry array
(2)    for each triangle:tri in spacegeom {
(3)        if (side1 <= mindst or side2 <= mindst or side3 <= mindst) then {
(4)            add tri into ArrayList<Geometry> can_tri
(5)        }
(6)    }
(7)    return can_tri

```

Procedure **getBridgingClusters**(ArrayList<Geometry> candidate_tri)

```

(1)    ArrayList<ArrayList<Geometry>> brdg_tri = new ArrayList<ArrayList<Geometry>>()//instantiation
(2)    Geometry ugeom = union candidate_tri
(3)    if (ugeom = "Polygon") then {
(4)        add candidate_tri into brdg_tri
(5)    }
(6)    else if (ugeom = "Multipolygon") then {
(7)        explode ugeom into separate polygons: ArrayList<Geometry> tricluster
(8)        add tri_cluster into brdg_tri

```

```

(9)      }
(10)     return brdg_tri

```

Procedure **bufferPlus**(Geometry geom, double dst)

```

(1)     BufferParameters bufp = new BufferParameters()
(2)     set bp.EndCapStyle(cap_square)
(3)     set bp.JoinStyle(flat_edge_corners)
(4)     BufferBuilder buffb = new BufferBuilder(bp)// instantiate buffer builder:buffb
(5)     Geometry buffg = buffb.buffer(geom,dst)
(6)     return buffg

```

Procedure **filterBridges**(ArrayList<ArrayList<Geometry>>>alg)

```

(1)     ArrayList<ArrayList<Geometry>>> falg = new ArrayList<ArrayList<Geometry>>>() //instantitation
(2)     if (alg.size() = 1) then {
(3)         if (alg.get(0).size() <= 2) then { //only a single or two triangle polygons exist
(4)             add alg.get(0) into falg
(5)         }
(6)         else { //more than one bridging gaps
(7)             ArrayList<Geometry> falgm = new ArrayList<Geometry>()//instantitation
(8)             Object[] robj = EdgeTrigraph(alg.get(0) //creating triangle edge graph
(9)             Map mapdat = Map<String,List> robj[0]
(10)            Map<Double,List> nmap = getSharedeEdgeTris(mapdat)
(11)            double edst = getMindstEdgeTriList(nmap)
(12)            List tripair = nmap.get(edst) //pair of triangles with the edge with min dst
(13)            tri_id1 = tripair.get(0)
(14)            tri_id2 = tripair.get(1)
(15)            Map nodgeom = Map<String, Geometry> robj[2]
(16)            Geometry tri_geom1 = nodgeom.get(tri_id1) //retrieving geometry for tri_id
(17)            Geometry tri_geom2 = nodgeom.get(tri_id2) //retrieving geometry for tri_id
(18)            //Adding two geometries into geometry array
(19)            add tri_geom1 into ArrayList<Geometry> :falgm
(20)            add tri_geom2 into ArrayList<Geometry> :falgm
(21)            add falgm into ArrayList<ArrayList<Geometry>>>:falg
(22)        }
(23)    }
(24)    else {
(25)        for (i=0, i < alg.size(), i++) {

```

```

(26)         ArrayList<Geometry> falgm = new ArrayList<Geometry>()//instantitation
(27)         if (alg.get(i).size() <= 2) then { //only a single or two triangle polygons exist
(28)             add alg.get(i) into falg
(29)         }
(30)         else
(31)         {
(32)             Object[] robj = EdgeTrigraph(alg.get(0) //crreating triangle edge graph
(33)             Map mapdat = Map<String,List> robj[0]
(34)             Map<Double,List> nmap = getSharedeEdgeTris(mapdat)
(35)             double edst = getMindstEdgeTriList(nmap)
(36)             List tripair = nmap.get(edst) //pair of triangles with the edge with min dst
(37)             tri_id1 = tripair.get(0)
(38)             tri_id2 = tripair.get(1)
(39)             Map nodgeom = Map<String, Geometry> robj[2]
(40)             Geometry tri_geom1 = nodgeom.get(tri_id1) //retrieving geometry
(41)             Geometry tri_geom2 = nodgeom.get(tri_id2) //retrieving geometry
(42)             //Adding two geometries into geometry array
(43)             add tri_geom1 into ArrayList<Geometry> :falgm
(44)             add tri_geom2 into ArrayList<Geometry> :falgm
(45)             add falgm into ArrayList<ArrayList<Geometry>>:falg
(46)         }
(47)     }
(48) }
(49) return falg

```

Procedure **EdgeTrigraph**(ArrayList<Geometry> alg)

```

(1)    //object instantiation
(2)    Object obj = new Object[4]
(3)    Map<String,List> hmp = new HashMap<String,List>()
(4)    Map<String,Geometry> igeom = new HashMap<Geometry,String>()
(5)    Map<Geometry,String> geomi = new HashMap<Geometry,String>()
(6)    for (i = 0, i < alg.size(), i++) {
(7)        Geometry tri_geom = alg.get(i)
(8)        normalize three line segments:ls1, ls2 and ls3 in tri_geom
(9)        String l1 = ls1_coordinates_start + ls1_coordinates_end //concatante coords
(10)       String l2 = ls2_coordinates_start + ls2_coordinates_end //concatante coords
(11)       String l3 = ls3_coordinates_start + ls3_coordinates_end //concatante coords

```

```

(12)         if (hmp.get(l1) = null) then
(13)             {
(14)                 List nl = new ArrayList()
(15)                 nl.add(i+1)
(16)                 hmp.put(l1, nl)
(17)             }
(18)         else
(19)             {
(20)                 if (hmp.get(l1).contains(i+1) = false)
(21)                     hmp.get(l1).add(i+1)
(22)             }
(23)         if (hmp.get(l2) = null) then
(24)             {
(25)                 List nl = new ArrayList()
(26)                 nl.add(i+1)
(27)                 hmp.put(l1, nl)
(28)             }
(29)         else
(30)             {
(31)                 if (hmp.get(l1).contains(i+1) = false)
(32)                     hmp.get(l1).add(i+1)
(33)             }
(34)         if (hmp.get(l3) = null) then
(35)             {
(36)                 List nl = new ArrayList()
(37)                 nl.add(i+1;
(38)                 hmp.put(l1, nl)
(39)             }
(40)         else
(41)             {
(42)                 if (hmp.get(l1).contains(i+1) = false)
(43)                     hmp.get(l1).add(i+1)
(44)             }
(45)     }//end for
(46)     obj[0] = hmp
(47)     obj[1] = igeom
(48)     obj[2] = geomi

```



```

(49)    obj[3] = alg.size()
(50)    return obj

```

Procedure **GetEdgeDistandTriIDs**(HashMap<String,List> map)

```

(1)    Map <Double,List>dmap = new HashMap<Double,List>();//object instantiation
(2)    Iterator it = map.entrySet().iterator
(3)    while (it.hasNext()) {
(4)        String key = (String)entry.getKey()
(5)        List value = (List)entry.getValue();// this consists of triangle ids
(6)        split coordinate pair in key
(7)        get the distance:edst between coordinate pair
(8)        if (value.size() > 1) then {
(9)            put edst and value in dmap
(10)    }
(11) }
(12)    return dmap

```

Procedure **GetMinDstEdgeandTriIDPair**(HashMap<Double,List> map)

```

(1)    Set set = map.keySet()
(2)    TreeSet treeset = new TreeSet()
(3)    add set to treeset
(4)    double mindst = treeset.first()
(5)    return mindst

```

G.9 Building aggregation with non-orthogonal sides

Input:

Building cluster (non-orthogonal) multipolygon geometry: cgeom

Minimum distance threshold between buildings: mindst

Minimum side length of a building: minlength

Space triangle edge distance threshold: sedgth

Inner hole area threshold: areath

Simplification threshold: simth

Output:

Building amalgam array: ArrayList<Geometry> bamal

Main method: NonorthogonalAmalgam(bclust, mindst, sedgth, minlength, fth, areath, simth)

```
(1)  ArrayList<String> blnk = new ArrayList<String>()
(2)  ArrayList<Geometry> triarray = new ArrayList<Geometry>()
(3)  ArrayList<Geometry>:bamal
(4)  int i = 0
(4)  for each cluster:cgeom in PostGIS database {
(5)      Geometry symbol = symbolization(cgeom)
(6)      if (symbol = null) then {
(7)          Geometry buffout = buffer_operation(cgeom,mindst,-mindst)
(8)          if (buffout.GeometryType = "Polygon") then {
(9)              if (buffout has inner holes) then {
(10)                  Geometry newg = IncludeInnerHoles(buffout, areath)
(11)                  remove inner vertices of newg: rnewg
(12)                  Geometry simpg = simplification(rnewg,simth)
(13)                  add simpg into ArrayList<Geometry> bamal
(14)              }
(15)          else {
(16)              remove inner vertices of newg: rnewg
(17)              Geometry simpg = simplification(rnewg,simth)
(18)              add simpg into ArrayList<Geometry> bamal
(19)          }
(20)      }
```

```

(21)         else if (buffout.GeometryType = "Multipolygon") then {
(22)             Geometry densgeom = Densifier.densify(buffout,mindst) //densification
(23)             explode buffout into separate polygons: ArrayList<Geometry> splitgeom
(24)             Object[] val4 = recreatingGeometry(splitgeom)
(25)             Map idgeom = val4[0]
(26)             Map geomid = val4[1]
(27)             Map clst = val4[2]
(28)             Map STRindx = val4[3]
(29)             clear blnk //empty building link array
(30)             clear triarray //empty building link array
(31)             ArrayList<Geometry> triarray = ConstrainedTriangulation(splitgeom)
(32)             get adjacency relations between each pair of buildings in blnk array
(33)             remove duplicate links in blnk
(34)             ArrayList<Geometry> spacetri = pairwise2candAggre(blnk, clst,
(35)                 STRindx, geomid, idgeom, mindst)
(36)             Object[] robj = EdgeTrigraph(spacetri)
(37)             Map mapdat = (Map<String,List>) robj[0] //edge vs tri_id list
(38)             Map geomnode = (Map<Geometry,String>) robj[1] // Geom vs tri_id
(39)             Map nodegeom = (Map<String,Geometry>) robj[2] // tri_id vs geom
(40)             Integer numt = (Integer) robj[3] //Getting Number of triangles
(41)             remove edges that share only single triangle id from mapdat
(42)             get adjacent triangle pairs: List<List<String>> adjpair from mapdat
(43)             sort adjacent triangle pairs in a sequence: List<List<String>> spair
(44)             get the sorted triangle ids: List<String> stri in a sequence from spair
(45)             get the topologically sorted trangle geometry:ArrayList<Geometry> aslg
(46)             //from spair and nodegeom
(47)             map trid vs geometry //triid -> dummy_id
(48)             map geometry vs trid
(49)             ArrayList<Geometry> ageompair = pairwisecandidateTriangles(aslg, clst,
(50)                 bid1, bid2, sedgth) //selected candidate tris based on edge distance
(51)             get the traingle ids:ArrayList<String> dummyids from ageompair in
(52)             // ascending order
(52)             check trids order (sequential or not) with boolean: chk
(53)             if (ageompair.size() = 1) then { //only one triangle
(54)                 if (first id of dummyids = 1) then {
(55)                     iterate adjoining triangles ascending until invert
(56)                     //triangle is found and add triangles to bamal array

```

```

(57)         }
(58)         if (first id of dummyids = ageompair(size) - 1) then {
(59)             iterate adjoining triangles descending until invert
(60)             //triangle is found and add triangles to bamal array
(61)         }
(62)     }
(63)     else if (chk = true) then {
(64)         if (ageompair.size() = 2) then {
(65)             pair dummyids
(66)             if (triangle_pair is invertible) then {
(67)                 add both triangles to bamal array
(68)             }
(69)             else {
(70)                 get two invertible triangles adjoining tri pair
(71)                 add triangle:trif with minimum area together
(72)                 // with initial triangulation pair to bamal array
(73)             }
(74)         else (
(75)             add all triangles to bamal array
(76)         )
(77)     }
(78)     else {
(79)         pair dummyids
(80)         get filling triangles of gaps between adjacent triangle clusters
(81)         //from both ends of each cluster using invert selection method
(82)         add filling triangles to bamal array
(83)     }
(84) }
(85)     }
(86)     else {
(87)         add Geometry: symbol to bamal array
(88)     }
(89) }
(90) return bamal

```

Procedure **IncludeInnerHoles**(Geometry geom, double ath)

- (1) Geometry result = null
- (2) **convert** geometry:geom into polygon:p
- (3) **create** the exterior ring:extrng of Polygon:p
- (4) **get** the number of inner rings:nring from polygon:p
- (5) **for** (i = 0, i < nring, i++) {
- (6) LineString lsring = **get** interior of nring(i)
- (7) **get** Coordinates[]:lscoord of lsring
- (8) **create** linear ring:lrhole from lscoord
- (9) **create** geometry:inggeom from lrhole
- (10) **if** (inggeom.getArea() >= ath) **then** {
- (11) lrrarray[i] = lrhole
- (12) }
- (13) }
- (14) result = new GeometryFactory().createPolygon(extrng, lrrarray)
- (15) **return** result

Procedure **pairwise2candAggre**(ArrayList<String> blink, Map<Coordinate, List<String> clst, STRtree indx, Map<Geometry,String> geomid, Map<String,Geometry> idgeom, double mindst)

- (1) ArrayList<Geometry> spaceTri = new ArrayList<Geometry>()//instantiate array geometry
- (2) **for** (i=0, i < blink.size(), i++) {
- (3) **get** bid1, bid2 and distance:dst between bid1 and bid2 from blink array
- (4) Geometry bldg1 = idgeom.get(bid1)
- (5) Geometry bldg2 = idgeom.get(bid2)
- (6) Geometry ugeom = **union** bldg1 and bldg2
- (7) **if** (dst <= mindst) **then** {
- (8) ArrayList<Geometry> alg = **ConstrainedTriangulation**(ugeom)
- (9) **for each** triangle: tri in alg {
- (10) **if** (tri_ver1 != tri_ver2) **then** {
- (11) add triangle:tri into spaceTri
- (12) }
- (13) }

```

(14)         }
(15)     }
(16)     return spaceTri

```

Procedure **pairwisecandidateTriangles**(ArrayList<Geometry> aslg, Map<Coordinate, List<String> clst, String bid1, String bid2, double thdst)

```

(1)     ArrayList<Geometry> calg = new ArrayList<Geometry>()//instantiate array geometry
(2)     for (i=0, i < aslg.size(), i++) {
(3)         Coordinate[] tric = aslg.get(i).getCoordinates()
(4)         get id lists lst1, lst2 and lst3 linked to each node of the triangle
(5)         get three edge distances: dst1, dst2 and dst3 in the triangle between bid1 and bid2
(6)         if (dst1 <= thdst or dst2 <= thdst or dst3 <= thdst) then {
(7)             calg.add(aslg.get(i))
(8)             create STRtree index of aslg.get(i)
(9)         }
(10)    }
(11)    return calg

```

H Terminology

A list of words often used in the dissertation is defined to avoid misunderstanding.

Attribute discretization

It is the transformation of numeric attributes into a smaller number of distinct ranges of nominal values.

Conforming Delaunay triangulation

It is the process of Delaunay stable mesh generation with a set of input planar points and vertices of Steiner points. In this process, splitting the triangles at the circumcentre and the edges that need to be served as constraints by adding Steiner points is adopted to generate a quality mesh by a refinement process with some shape criteria applied to triangles. As a result of splitting triangles and edges, the Steiner points may be added to the interior, the boundary and outside the boundary of the convex hull of the input planar point set.

Constrained Delaunay triangulation

It is the process of generalizing the conventional Delaunay triangulation with constraining a set of planar edges among a set of input planar points, respecting no longer the empty circle criterion adopted in the Delaunay triangulation. As a result, this triangulation is not Delaunay stable.

Delaunay constrained triangulation

It is the process of generating Delaunay triangulation with constraining a set of planar edges among a set of input planar points.

Delaunay triangulation

It is the most optimized triangulation in terms of the shape of triangles such that the triangulation maximizes the minimum angle of its triangles. When the triangulation is Delaunay, it has the property that no point in a planar input point set is inside the circumcircle of any triangle (empty circumcircle criterion).

Densification of edges

It is the process of splitting existing planar line segments into several sub-segments at regular intervals.

Granular edges

Detail edges of a building geometry formed due to the application of squaring the edges followed by the enlargement of a building in the automatic map generalization.

Polygon triangulation

It is the process of decomposition of a simple polygon into triangles. The definition of a simple polygon is that it does not have any vertex shared by more than two edges nor does it have new vertices created by the intersection of two non-consecutive edges of the polygon.

Steiner point

It is a point with a particular geometric relation to a triangle, although it is not part of the input set of planar points used to generate triangulation.

Steiner triangulation

It is the process of deriving a triangular mesh with some optimal shape criteria in generating triangles by adding Steiner points. In this process, such Steiner points may be added to the interior, the boundary and outside the boundary of the convex hull of the input point set.